Applications of Nonlinear Dynamics to Information Processing

Zoran Aleksić

A thesis submitted for the degree of Doctor of Philosophy at the Australian National University

May 1990
The work presented in this thesis is my own except for results of the fourth chapter which were obtained working together with Dr. Stjepan Marčelja. None of the work reported here has been submitted to any other institution of learning for any degree.

The shorter version of the second chapter was submitted for publication to Physica D. The results of the third chapter will be submitted to Physics Review A, and the results of the fourth chapter will be submitted to Biological Cybernetics.

Zoran Aleksić
Acknowledgements

I would like to thank my supervisor Stjepan Marčelja for teaching me what science really is and his patience and tolerance.

I thank Adrian Horridge for providing a nice environment for studying and generously allowing me to undertake research which was not directly connected with the activity of the Department.

I thank Terry Bossomaier, Daniel Osorio, Andrew James and Geoff Stuart for many helpful discussions.
ABSTRACT

The reported results are direct applications of nonlinear dynamics to information processing or are relevant for the applications. In the second chapter we describe a simple method for estimating the embedding dimension that can be used as a first step in constructing nonlinear models. The method for the reduction of measurement noise in chaotic systems that is presented in the third chapter is attractive in the cases where high accuracy is necessary. Next we propose how to overcome some problems encountered in constructing models of complex nonlinear systems. Finally, the behaviour of one—dimensional cellular automata useful for the detection of velocities of patterns is shown and explained in the last chapter.

The method of estimating the embedding dimension is based on the idea that when the observed dynamical system is deterministic and smooth and the embedding dimension is correctly chosen, the relationship between the successive reconstructed state vectors should be described as a continuous mapping. To check if the given embedding dimension is a good one we search for pairs of state vectors whose distance is smaller than some number. For each pair we compute the distance between the successors of the elements of pairs and represent this distance graphically. When the embedding dimension is equal or larger than the minimum correct dimension, all distances are small in comparison to distances for incorrect dimensions.

The method for noise reduction is developed assuming that the map of the system is known and the noise is bounded. The closer the initial condition is to the true state of the system, the longer the computed trajectory follows the observed trajectory. To reduce the uncertainty in knowing the given state we recursively search for the state for which the computed trajectory follows the observed trajectory as long as possible. The method is demonstrated on several two—dimensional invertible and noninvertible chaotic maps. When the map is known exactly an arbitrary level of noise reduction can be achieved.
With the increase of the complexity of a nonlinear system it is harder to construct its model. We propose to discover first how to construct a model of a similar but simple system. Discovered heuristics can be useful in modeling more complex systems. We demonstrate the approach by constructing a deterministic feed-forward neural network that can extract velocities of one-dimensional patterns. Analyzing simpler models we discovered how to estimate the necessary numbers of neurons; what are the useful ranges of the parameters of the network and what are the potential functional dependencies between the parameters.

The class of one-dimensional cellular automata whose state is a function of both the previous state and a time—dependant input is described. As inputs we considered the sequences of binary strings that represent black—and—white objects moving in front of a white background. As outputs we considered the trajectory of the automaton. For some rules the automaton will evolve to the zero state for all velocities of the object except for the velocities in specific narrow range. The phenomenon is persistent even when a strong noise is present in input patterns but unreliable units of the automaton or having a more complex input break it down.
CONTENTS

1 Introduction 8
   1.1 Information processing as nonlinear dynamics 9
   1.2 Adaptation as nonlinear dynamics 10
   1.3 Nonlinear model-based processing 12
   1.4 Nonlinear dynamics of distributed processing systems 13
   1.5 The organisation of the thesis 14

2 Estimating the embedding dimension 17
   2.1 State space reconstruction 19
   2.2 Previous approaches 24
   2.3 A criterion and a procedure for estimating the embedding dimension 25
   2.4 Numerical experiments 30
   2.5 Relationship of the proposed approach to some previous ones 39

3 Noise reduction in chaotic systems 41
   3.1 Chaos 'creates' information 42
   3.2 Measuring the information acquisition rate 44
   3.3 Previous approaches to noise reduction 46
   3.4 Determining a state with improved precision by maximising
       the duration of correct predictions 48
   3.5 Two-dimensional invertible mapping 53
   3.6 Two-dimensional noninvertible mapping 58
   3.7 The influence of the strength of noise and uncertainty
       in knowing the map 61
   3.8 Further improvements and discussion 62
4 Discovering heuristics for constructing neural networks 63
  4.1 Feed-forward neural network 64
  4.2 Determining the parameters of the network 66
  4.3 Bad topology of the objective function and the problem of scale 68
  4.4 Heuristic search 70
  4.5 Constructing a small network 73
  4.6 The analysis of the successful configurations 75
  4.7 Constructing large network using heuristic rules 80
  4.8 Discussion 83

5 Some aspects of processing temporal inputs with cellular automata 84
  5.1 Cellular automata 86
  5.2 One-dimensional cellular automata as filters of binary strings 89
  5.3 Processing temporal inputs with the cellular automata 91
  5.4 Filtering of the velocities 94
  5.5 The mechanism of filtering 97
  5.6 Breakdown of filtering 99
  5.7 Conclusion 102

REFERENCES 103
chapter 1

Introduction

Today's information processing systems are much faster, more reliable and cheaper than systems built only a decade ago. However, their capabilities for performing some tasks are still not satisfactory. To perform tasks of higher complexity they need extensive and laborious programming. Small changes of the task may mean starting their programming from scratch. In some tasks like recognising extended speech patterns or extracting information from images, good performance has not been achieved. The tasks where artificial information processing systems perform badly are usually the tasks in which natural systems are very good. Thus, tentatively, one might use the principles of information processing by natural systems for constructing complex artificial systems. The only problem is that many of these principles still wait to be discovered. We know that natural systems perform some complex tasks well but we do not know how.
Recent advances in theory of nonlinear dynamics reveal that even simple systems with only a few degrees of freedom can exhibit complex behaviour. On the other hand, the evolution of a system with many degrees of freedom can sometimes be modelled by a system of few degrees of freedom. The complexity may result from the repetition of simple nonlinear transitions. Such results, which throw more light on the complex behaviour, suggest applying nonlinear dynamics to information processing. They may be useful both for understanding natural processing systems and for constructing complex artificial systems.

During our PhD study we attacked different problems. The common feature of our approaches is that they are either the applications of nonlinear dynamics to information processing or relevant for such applications. During the past few years, the theory of nonlinear dynamics was broadly applied to different areas of information processing. In the next four sections we will sketch some aspects of the application of nonlinear dynamics to information processing. Our purpose is not to give an extensive list of possible applications, but to provide a perspective for the results presented in the thesis.

1.1 Information processing as nonlinear dynamics

The area that has received the most attention to date is the one where an information processing system is viewed as a nonlinear dynamical system. In general, processing produces outputs that depend in some desired way on given inputs [Deutsch, 1989]. Regardless of its content, it is a physical process, and a processing system is a physical object. The initial state of a nonlinear dynamical system can be taken as an input. A motion of the system is regarded as the performance of a
processing. The evolution of the system is governed by a nonlinear differential equation of the general form

\[
\frac{dx}{dt} = F(x(t), a), \quad (1.1a)
\]

or a nonlinear mapping

\[
x(n + 1) = f(x(n), a), \quad (1.1b)
\]

where \( x \) is the state of the system, and the components of \( a \) are the parameters of the differential equation or mapping. In most applications, the attractors of the system are regarded as the outputs and the approach is called ‘computing with attractors’ [Huberman, 1985]. The system can be embodied as a set of interconnected analog amplifiers [Tank and Hopfield, 1986] or a set of simple digital processors [Huberman, 1985], or indeed as any system whose evolution can be described with equations (1.1). For example in the case of a neural network [Amari, 1972] the differential equation (1.1a) takes the form

\[
\frac{dx_i}{dt} = -x_i + \phi\left( \sum_j w_{ij} x_j \right), \quad (1.2)
\]

where \( x_i \) represents the activity of the \( i \)-th neuron, \( w_{ij} \) is the connection strength between two neurons and \( \phi \) is a monotonically increasing continuous function.

**1.2 Adaptation as nonlinear dynamics**

The process of adaptation is a successive modification of the structure of a system towards better performance. It is common in diverse fields like economy,
psychology, or biology [Holland, 1975]. The adaptation is a very complicated motion in the parameter space. For example, Farmer et al. [1986] described the behaviour of the immune system and learning in a simple artificial system as a nonlinear dynamical system. Complex artificial systems are the result of a long process of adaptation. Although the adaptation is one of the least understood types of nonlinear dynamics, even a rudimentary knowledge is useful for the construction of artificial systems.

Let us illustrate a simple process of adaptation. Suppose we want to determine the connection strength $w_{ij}$ so that the system’s (1.2) trajectory between $t_0$ and $t_1$ starting at $x(t_0)$ is as close as possible to the desired trajectory $s(t)$. This can be accomplished by minimising a functional

$$E = \int_{t_0}^{t_1} \| x(t) - s(t) \| dt,$$

(1.3)

where the norm may take several forms. The connection parameters of the system $w_{ij}$ are modified according to the equation [Pearlmutter, 1989]

$$\frac{dw_{ij}}{dt^*} = -\eta \frac{\partial E}{\partial w_{ij}},$$

(1.4)

where $\eta$ is a constant, and $t^*$ is time of the adaptive process. When the norm is Euclidian, then [Pearlmutter, 1989]

$$\frac{\partial E}{\partial w_{ij}} = \int_{t_0}^{t_1} x_i \phi' \left( \sum_j w_{ij} x_j \right) z_j dt;$$

(1.5)

where $z_i$ are defined by the differential equation
with the boundary condition \( z_i(t_1) = 0 \), and \( \phi' \) is derivative of \( \phi \). In this example, one is interested in the fixed points of the equation (1.4) that are potentially successful configurations of the system (1.2).

1.3 Nonlinear model—based processing

Very often complex information processing is model—based. For example, in speech recognition words can be represented by hidden Markov models [Rabiner and Juang, 1985]. To recognise a given word is to find from the set of models the one that gives the best description of the word. Very efficient coding of face images can be achieved by constructing 3—D facial model and using the model’s parameters as the code [Aizawa et al. 1989]. Noise in the input data can be effectively reduced when the model of data is at least partially known [Gull and Daniell, 1978]. Knowledge of the model of the system is the essential condition for the prediction of the system’s evolution [Rosen, 1985].

In some cases the input can be modelled successfully as a trajectory of a nonlinear dynamical system. When the inputs are temporal signals, the approach is straightforward. The sequence of elements

\[
s_1, s_2, \ldots, s_n
\]  

(1.7)
can be viewed as the evolution of the dynamical system

\[ s_{i+1} = f(s_i, a). \]  

(1.8)

Constructing a nonlinear model of the input means determining the parameters \( a \) for which the mapping \( f \) approximates sufficiently well the relationship between the successive elements. Available techniques for representing functional relationships (1.8) are diverse and range from local approximation [Farmer and Sidorowich, 1988], tesselation of data points [Mees, 1989] to neural networks [Lapades and Farber, 1987].

Recently, Farmer and Sidorowich [1988] achieved good results in forecasting and noise reduction using a nonlinear model. Lapedes and Farber [1987] showed that nonlinear models are useful for system engineering. The method for constructing nonlinear models of systems like cellular automata developed by Meyer et al. [1989] may be useful for pattern recognition.

1.4 Nonlinear dynamics of distributed processing systems

Ultimately information processing is always a temporal process. While the theory of nonlinear dynamics is usually not applied to the analysis of conventional computers, it can give some insights about the behaviour of distributed processing systems. With the advance of technology it is now feasible to build large-scale distributed processing systems but it is still unclear how to organise processing or how to predict the behaviour of a particular configuration. Large-scale distributed systems share some common properties [Hewit and de Jong, 1982]: they are continuously evolving; the decision making mechanism is decentralised; the distributed processors do not have access to one another's internal information; the
information is exchanged with delays; and the information may become inconsistent among different processors.

Recently attempts were made to apply theory of nonlinear dynamics to the understanding one such system [Huberman and Hogg, 1988; Kephart et al., 1989]. Their approach is to model a system so that the obtained equations have the form of the equations of nonlinear dynamics and to draw conclusions based on the analogy. For example, the behaviour of cooperating processors choosing between two strategies and communicating with a delay can be described as

\[ q(t + 1) = \frac{1 + \text{erf}(\Delta + \Lambda q(t) - \Psi q^2(t))}{2} \]  

where \( \Delta, \Lambda, \) and \( \Psi \) are constants, \( q \) is the proportion of processors using the first strategy, and \( \text{erf}(x) \) is the error function. For a wide range of parameter values this equation belongs to a universal class of nonlinear dynamical systems [Collet and Eckmann, 1980], therefore for some values of the parameters the behaviour of the system can be chaotic. When constructing a real system this knowledge can be used for choosing systems parameters. The research in this area scarcely started and it will be interesting to see how useful the approach turns out to be.

1.5 The organisation of the thesis

Three chapters of the thesis present the results concerned with nonlinear model—based processing. The second chapter shows an approach to estimating the embedding dimension. Very often the data are incomplete. In order to build a nonlinear model from such data the state vectors have to be reconstructed. The dimension of reconstructed state vectors is called the embedding dimension. When the incomplete data are obtained observing a deterministic smooth system and the
state vectors are well reconstructed, the relationship between successive state vectors should be described as a continuous mapping. Our approach consists of searching for the lowest embedding dimension for which the property of a continuous mapping is satisfied: the images of close points are close. We demonstrate the effectiveness of the approach by finding the lowest embedding dimension of several chaotic systems embedded by the method of delays.

Knowledge of the nonlinear model of a chaotic system can provide a substantial reduction of measurement noise. In the third chapter we describe how to determine a state of a chaotic system with high precision from a sequence of unprecise measurements. We assumed that the model of the system is known. The procedure we are using is essentially the optimisation process. The better we determine a given state, the longer will the computed trajectory starting from this state follow the observed trajectory. We search for the state for which the computed trajectory stays close to the observed trajectory as long as possible. When the model was known exactly, we achieved an arbitrary accuracy in determining the states for several one—and two—dimensional systems we tested.

Nonlinear models can be constructed using neural networks. Parameters of the network are usually determined by the process of adaptation (training). In constructing more complex models the available methods of adaptation usually encounter problems. In the fourth chapter we describe a solution to one such problem. Like artificial intelligence approach to solving problems, we first construct small networks, analyse successful configurations and use the obtained knowledge to construct large networks. The approach is demonstrated constructing a deterministic feed—forward neural network that can extract velocities of one—dimensional patterns.
In the last chapter we consider information processing as the evolution of a nonlinear dynamical system. The system we are using for processing is a one—dimensional cellular automaton. We describe a class of automata that are continuously interacting with time—dependent inputs. The processing with such a system can be viewed as a mapping from an input trajectory to the trajectory of the system. We take as input a sequence of binary strings that can be visualised as black—and—white objects moving in front of a white background. In some cases, the trajectory of the automaton is sharply dependent of the velocity on the object. The automaton will not evolve into the zero state only when the velocities are within a narrow range. The phenomenon can be destroyed by having unreliable components or adding more complicated input.
The data for constructing nonlinear models are seldom complete. For example, it is practically impossible to obtain detailed information about the turbulent behaviour in a fluid flow. Speech is produced by a rather complicated mechanism but all information for speech recognition is contained in the time variation of sound pressure. In some cases, partial data may be sufficient for building qualitatively equivalent model, which can even provide short time prediction.

When the data are not complete, the dynamics of the observed system has to be reconstructed. The number of components of the reconstructed state vectors is called the embedding dimension. In order to achieve a faithful reconstruction, the embedding dimension has to be correctly chosen. Its value is connected with the number of degrees of freedom of the observed dynamics but usually there is no a priori knowledge about the observed system. The only way to estimate the embedding dimension is to analyse available data.
The approach to estimating the embedding dimension proposed in this chapter is very simple. We search for the minimum embedding dimension for which the reconstructed dynamics can be described as a continuous mapping. Starting with lower embedding dimension we calculate the distances between the images of close points and represent them graphically. When the minimum embedding dimension is reached all distances suddenly become small and decrease very slowly with increasing dimensions.

The approach can be implemented efficiently. Moreover, the largest part of the algorithm, finding the close neighbours, is commonly available. For example, it can be already found in programmes for calculating Liapunov exponents [Eckmann et al., 1986] or constructing the predictive model [Farmer and Sidorowich, 1988]. The proposed approach is in some cases not as accurate as some previous approaches but because of its simplicity and efficiency we believe that it should be useful as a first step in assessing an unknown dynamical system. The information extracted using this approach can be later used to avoid unnecessary computation when more accurate approaches are applied. While we tested the proposed approach in examples of reconstruction by delay methods, it is applicable to other methods as well.

In the first section of this chapter we will describe methods for state space reconstruction. Previous approaches to estimating the embedding dimension will be reviewed in the following section. Our approach and its numerical investigations will be described in sections 2.3 and 2.4. In the last section we will compare our approach with some previous approaches.

---

1 When there are not many data points, the approaches based on the prediction may be superior.
2.1 State space reconstruction

We will assume that data to be used for building a nonlinear model are obtained by observing a dynamical system that can be described as a differential equation

\[
\frac{dx(t)}{dt} = F(x(t)),
\]

(2.1)

or as a mapping (discrete-time case)

\[
x(n + 1) = f(x(n)),
\]

(2.2)

where \( F \) or \( f \) are differential functions, and \( x = <x_1, x_2, \ldots > \) is a finite or an infinite state vector. Equations (2.1) and (2.2) can describe a flow of a viscous fluid [Ruelle, 1989], the dynamics of a population density [May, 1975], or a chemical reaction [Berge et al., 1986]. The states of the system may be thought of as points in state space or a phase space. The state space can be a \( \mathbb{R}^n \) space, or a manifold like a sphere or a torus, or an infinite dimensional Hilbert space.

When observing real systems, we seldom have access to all variables. Very often one is monitoring only one scalar variable. We will assume that the obtained data are in the form of a time series of scalar values

\[
s(t), \ s(t + \tau), \ s(t + 2 \tau), \ldots, \ s(t + N \tau)
\]

(2.3)

where the samples are taken at constant intervals \( \tau \) of time. The extension to the cases where the elements of the time series are vectors is straightforward. The elements of time series are a smooth function of system's states (an observable) \( s(t) = \mathcal{U}(x(t)) \).

The first step in analysing partial data is the reconstruction of state space. The heuristic idea behind the reconstruction is that the elements of time series should be combined in such way that the observed system is uniquely labeled at any moment of time. The
relationship between the successive reconstructed states should be described as equations (2.1) and (2.2). The dimension of reconstructed states is usually called the embedding dimension, but the term 'reconstruction dimension' is used as well [Fraser, 1989a].

\[ \mathbf{M} \rightarrow x(t) \rightarrow \mathbf{Y}^{-1} \rightarrow \mathbf{M}' \]

**Figure 2.1** A schematic representation of the process of reconstruction. The observation of the dynamical system \( f \) provides a time series \( s(t) \). The system's states are reconstructed as \( m \)-dimensional vectors. The relationship between the true phase space and the reconstructed one is an embedding \( \Psi \).

The process of the reconstruction of state vectors is illustrated in Fig. 2.1. The observation of the dynamical system \( f \) provides a time series \( s(t) \). The reconstructed state vectors are points in \( m \)-dimensional phase space. Partially observed dynamical
system and its good reconstruction are differentially equivalent [Takens, 1981]. The relation between the points on the observed and reconstructed orbits is one-to-one. The closed orbits are reconstructed as closed. The singularities of the reconstructed system correspond to the singularities of the observed system. A relationship between the observed phase space and its reconstruction can be mathematically described as an embedding. The embedding is a smooth mapping that is a diffeomorphism (one to one differentiable and one to one differentiable inverse) from the manifold to a submanifold of \( \mathbb{R}^m \) space, [Chillingworth, 1976].

The easiest way to reconstruct state vectors \( y(t) = < y_1(t), ..., y_m(t) > \) at time \( t \) is to take as its components several ‘delayed’ elements of time series

\[
\begin{align*}
y_1(t) &= s(t), \\
y_2(t) &= s(t + \Delta), \\
&\quad \vdots \\
y_m(t) &= s(t + (m - 1) \Delta),
\end{align*}
\]

(2.4)

where \( m \) is the embedding dimension, and the delay \( \Delta \) is a multiple of the sampling interval. This method of reconstructing state vectors is called the method of delays and it was proposed by Ruelle [Packard et al., 1980]. Another way to reconstruct state vectors is to estimate time derivatives of the experimental signal and take as components of reconstructed state

\[
y_k(t) = \frac{d^k s(t)}{dt^k}.
\]

(2.5)

This approach to reconstruction is feasible if the sampling time is sufficiently small. The measurement noise is amplified with the differentiation so this method is not practical for reconstruction of states with higher dimensions. Broomhead and King [1986]
proposed to take as components of reconstructed state linear combination of delays

\[ y_k(t) = \sum_i c_{ki} s(t + i \Delta) \]  \hspace{1cm} (2.6)

where the \( c_{ki} \) coefficients are determined by Karhunen and Loeve principal value decomposition. Nice features of this approach are that it has a built-in filter for reducing noise and the reconstructed dynamics is easier to fit. But as pointed out by Mees \textit{et al.} [1987] and Fraser [1989b] it can produce bad reconstructions.

In the example given by Fraser [1989b] the best reconstruction was achieved by heuristically found nonlinear combination of delayed components. It is interesting to note that Hinton and McClelland [1988] proposed a nonlinear equivalent to principal value decomposition\(^2\). Their objective was to study internal representation in neural networks\(^3\). They trained feed-forward network to reproduce input at output by passing input through smaller number of hidden units then the dimensionality of input. This approach may retain the good features of Broomhead and King approach without being restricted with the linear independence.

Let us illustrate the reconstruction by the delay method using the time series obtained by the numerical integration of the Rössler system [Rössler, 1976]

\[
\begin{align*}
\dot{x} &= -(y + z), \\
\dot{y} &= x + 0.2y, \\
\dot{z} &= 0.4 + x y - 5.7 z.
\end{align*}
\]  \hspace{1cm} (2.7)

\(^2\) We did not check extensively in the literature if there is some previous work on nonlinear equivalents to principal value decomposition.

\(^3\) As we will mention in the fourth chapter, we view neural networks as one class of nonlinear mappings. The methods developed for the construction (training) neural networks can be used for the approximation with other basis functions as well.
The \((x(t), y(t))\) projection of the strange attractor of the Rossler system is presented in Fig. 2.2a. The reconstruction \((x(t), x(t-\Delta))\) is shown in Fig. 2.2b. The true attractor and the reconstructed one are topologically equivalent [Takens, 1981].

Whitney [Chillingworth, 1976] proved that every k-dimensional manifold can be embedded in 2k dimensional space. Takens [1981] proved that the relationship between the state space reconstructed by the methods of delay or by differentiation and true state space is an embedding when the following conditions are satisfied: \(m \geq 2k + 1\), where \(k\) is the dimension of the observed manifold; the mapping \(f\) is a smooth diffeomorphism; no two different fixed points of \(f\) are at the same level of \(s(t)\); and if \(x\) is a point with a period \(p, p \leq 2k + 1\), then all eigenvalues of the Jacobian at \(x\) are distinct and different from 1. Whitney and Takens results are not of use in choosing the correct embedding dimension because the dimension of the system is not known before the reconstruction. Furthermore, the embedding dimension 2k is
not necessarily the minimum one. In many particular cases, it is possible to embed a
given dynamical system in a space whose dimension is lower than $2k$.

### 2.2 Previous approaches

Several approaches to estimating the embedding dimension have been proposed.
Packard et al. [1980] proposed that the width of conditional probability distribution
$P ( s(t + \Delta) \mid s(t - \Delta), ..., s(t - (m - 1) \Delta) )$ indicates correct embedding
dimension. When the embedding dimension is large enough, the width of the distribution
is small. In the approach proposed by Cremers and Hübler (1987) the first step is to
estimate flow vectors $v(r)$ at points $r$ of reconstructed phase space. When the
embedding dimension is correctly chosen, the difference of two flow vectors $v(r_1)$ and
$v(r_2)$ at neighbouring points $r_1$ and $r_2$ must vanish if the distance between $r_1$ and $r_2$
vanishes

$$\lim_{r_1 \to r_2} \| v(r_1) - v(r_2) \| = 0 \quad (2.8)$$

To check if (2.8) holds for the chosen embedding dimension they take the average over
all flow vectors whose distance is closer than $\varepsilon$

$$W(\varepsilon) = \| v(r_i) - v(r_j) \| \quad , \text{ where } \| r_i - r_j \| < \varepsilon . \quad (2.9)$$

They shown empirically that the average $W(\varepsilon)$ scales linearly with $\varepsilon$ if the embedding
dimension is correct. Crutchfield and McNamara [1987] and recently Casdagli [1989]
proposed that deducing the optimal predictive model leads to an estimate of the minimum
embedding dimension. For Casdagli the predictive model is optimal if its predictive error
is as small as possible. Crutchfield and McNamara search not only for the model that
has small predictive error but that is as simple as possible as well. We will latter discus
the relationship between these three approaches we mentioned and our proposal for estimating the embedding dimension.

Froehling et al. [1980] proposed a method for determining the approximate fractal dimension of the chaotic flow, but their method also indicates the correct embedding dimension. They search for the embedding dimension for which the reconstructed attractor can be locally approximated by its tangent. Broomhead et al. [1987] approach is based on essentially the same idea but they use principal value decomposition that provides better numerical efficiency. Fraser's approach [1989a] is based on the minimisation of mutual information between the components of the reconstructed state vectors. Eckmann and Ruelle [1985] suggested that the goodness of a linear fit of the rate of short time divergencies of orbits indicates whether the embedding dimension has been correctly selected. In the approach proposed by Mees [1989], the construction of an optimal tessellation of sample points indicates the correct dimension.

2.3 A criterion and a procedure for estimating the embedding dimension

We assumed that the observed system is deterministic: future states are uniquely determined by the past ones. The observed system and its good reconstruction are diffeomorphic. Therefore, a well reconstructed dynamics is deterministic and should be described as a single-valued mapping

$$g : M' \rightarrow M', \quad (2.10)$$

of the space $M'$ into itself. The space $M'$ is the reconstructed phase space. The
mapping $g$ describes the relation between two successive reconstructed states

$$y(t + \tau) = g(y(t)),$$  \hspace{1cm} (2.11)

where $y(t) \in M'$ is the state of the system at time $t$. When the dynamics is well reconstructed, every state of the system at time $t$, $y(t) \in M'$, is followed by the unique state $y(t + \tau)$ after the time interval $\tau$. In other words, one cannot find two different reconstructed states $y_i(t + \tau)$, $(i = 1, 2)$, at $t + \tau$ assigned to the same state $y(t)$ at $t$.

$$\text{if } y_1(t + \tau) = g(y(t)) \text{ and } y_2(t + \tau) = g(y(t))$$

$$\text{then } y_1(t + \tau) = y_2(t + \tau)$$  \hspace{1cm} (2.12)

The property (2.12) is a part of the mathematical definition of the mapping. For time-continuous dynamical systems the property (2.12) is equivalent to the property of non-intersection of phase trajectories. The property (2.12) implies the small width of the conditional probability distribution, $P(y(t + \tau) \mid y(t))$ [Packard et al., 1980]. When the dynamics is reconstructed in the space whose dimension is equal or larger than the minimal correct embedding dimension, (2.12) is true for every point in phase space.

Let us illustrate the property (2.12). Suppose that we reconstructed state vectors and want to fit corresponding mapping. In Fig.2.3a is graphically represented relationship $x(n+1) = g(x(n))$ between successive points obtained by iterating the logistic map with the control parameter $\eta = 3.9$ [May, 1975]

$$x(n+1) = \eta \cdot x(n) \left( 1 - x(n) \right).$$  \hspace{1cm} (2.13)
Figure 2.3 The points $x(n+1)$ are plotted against points $x(n)$ for a) the logistic map (2.13) and b) the Hénon map (2.14).

It can be seen that there is no problem in fitting the corresponding map with a curve. The relation (2.12) holds at each points of phase space, therefore the correct embedding dimension is 1. The same is not true for the relationship $x(n+1) = g(x(n))$ between the points obtained by iterating the Hénon map [Hénon, 1976]

\[
x(n+1) = 1 + y(n) - 1.4 \, x(n) \\
y(n+1) = 0.3 \, x(n)
\] (2.14)

The correct embedding dimension for this system is 2. The relationship $x(n+1) = g(x(n), x(n-1))$ is represented in Fig.2.4a. The points are lying on the surface represented in Fig.2.4b. This illustration also shows why optimal predictive model indicates the minimum embedding dimension\(^4\). For instance, if the predictive error drops for the embedding dimension 2 that means that almost all points are lying on the surface.

\[^4\text{We assumed that the class of mapping used for fitting is complex enough to approximate given data.}\]
Figure 2.4  a) The relationship $x(n+1) = g(x(n), x(n-1))$ represented graphically for the points $x(n)$ obtained by iterating the Henon map (2.14).  b) The points are laying on the surface.

As it stands, the property (2.12) can be a criterion for choosing the embedding dimension but it is not very practical. It can be reformulated by noting that the available methods of reconstruction are an embedding only if the observed system is a smooth diffeomorphism [Takens, 1981]. Therefore, if the relation between the successive
reconstructed states is a mapping, that is (2.12) is true, that mapping is a continuous
one. When the mapping is continuous, the distances between the images of the close
points are small. For every two points \( y_i \) and \( y_j \) whose distance is smaller then \( \varepsilon \) one
can find that the distance between their images is smaller then some \( \delta \),

\[
\text{if } \| y_i - y_j \| < \varepsilon \text{ then } \| g(y_i) - g(y_j) \| < \delta .
\]  

(2.15)

For small \( \varepsilon \) the value of \( \delta \) will be proportional to \( \varepsilon \) and the slope of the mapping \( g \)
around \( y_i \). For one-dimensional mapping, \( \delta \) will be

\[
\delta = |\varepsilon \cdot g'(y_i)| .
\]  

(2.16)

where \( g' \) denotes the derivative of \( g \). The property (2.15) is easily checked. Our
approach to estimating the embedding dimension is to check how large are the distances
between the images of close points. When the given embedding dimension is equal to or
larger than the minimum correct one, all distances are small. In practice, it is not
necessary to know the slope of the mapping to decide if the distances between the
images are small. For all the reconstructions we tested, the distances drop sharply when
the trial embedding dimension reaches the minimum correct one. With further increase
of the embedding dimension the distances decrease more slowly.

Our approach to estimating the embedding dimension consists of the following
procedure. For a given embedding dimension we reconstruct state-vectors. The
reconstructed trajectory is in the form of time series \( \{ y_i \} \), \( i = 1, \ldots, N - m + 1 \). For
each point \( y_i \) we determine the points \( y_j \) which are closer then some distance \( r \)

\[
\| y_i - y_j \| \leq r .
\]  

(2.17)
For the points $y_j$, which are neighbours of $y_i$, we calculate the distances between the images of $y_i$ and $y_j$

$$d_{ij} = \| y_{i+1} - y_{j+1} \|. \quad (2.18)$$

We represent the computed distances $d_{ij}$ as the points on the plane parametrised by the first component of the reconstructed state vector as the horizontal axis and $d_{ij}$ as the vertical axis. Comparing the graphical representations of distances for different embedding dimensions we select the correct dimension.

The main reason for the graphical presentation of the calculated distances $d_{ij}$ was the fact that for many systems, when the trial embedding dimension is not the correct one, the distance between the images of close points may be large in only a small part of the reconstructed phase space. Any form of simple averaging of the distances over the whole phase space would dilute the useful information and make it more difficult to draw conclusions. Of course, one could conceive of a complicated algorithm that would detect the point in phase space where the distance is large and then analyse the mapping around that point more carefully. Such an algorithm would not be as fast, and would require more detailed knowledge about the mapping.

### 2.4 Numerical Experiments

We estimated the embedding dimension of several model chaotic systems. A system was iterated or integrated and one variable, usually $x$, was recorded in integer format. The

5 Actually such system would be a kind of expert system with a knowledge base about the nonlinear mappings.
integer format is chosen for computational efficiency. We start with a time series

\[ s_1, s_2, \ldots, s_N \] (2.19)

of \( N \) scalar data points sampled at constant time intervals or corresponding to each iteration. The process of observation of a physical system is seldom ideal. The measured signals are usually noisy. To take into account this fact we added uniform noise to the time series. The state vectors were reconstructed by the method of time-delay coordinates. We assume that the delay time is equal to the sampling time. The components of the reconstructed state vectors are several successive points

\[ y_i = < s_i, s_{i+1}, \ldots, s_{i+m-1} >, \] (2.20)

where \( m \) is the embedding dimension. We are using the \( L_\infty \) norm

\[ ||y_i - y_j|| = \max \{ |s_{i+k} - s_{j+k}| \} \] (2.21)

in order to speed up the computations [Eckmann et al., 1986]. The vectors \( y_i \) and \( y_j \) are closer than some small \( r \), therefore the absolute difference between the first \( m-1 \) components of \( y_{i+1} \) and \( y_{j+1} \) vectors is smaller than \( r \), as well. Because we are interested in distances larger than \( r \) the distances between the images (2.21) can be approximated as

\[ d_{i,j} = |s_{i+m} - s_{j+m}|. \] (2.22)

The distance \( d_{i,j} \) is plotted\(^6\) against \( s_i \).

---

\(^6\) If we reconstructed state vectors as \( y_i = < s_i, s_{i-1}, \ldots, s_{i-m+1} > \) the corresponding graphs would have different form for \( m > 1 \), but the results in estimation would be equivalent.
**Figure 2.5** The distances between the images of close points $d_{i,j}$ generated by the Hénon map, are plotted against $s_i$. In this figure horizontal and vertical axes have the same scale. The value of $r$ is 0.15\% of the extent of the series, uniform noise with maximum amplitude 1\% is added to the generated time series, and the number of samples $N$ is $5 \cdot 10^3$. 

A result for the Hénon system (2.13) is represented in Fig. 2.5. Uniform noise, with maximum amplitude 1\% of the maximal amplitude of the time series is added. The maximum neighbour distance, $r$, was 0.15\% of the amplitude. When the embedding
dimension $m$ is 1, the distances of images of close points are up to one third of the series amplitude. For dimensions greater than 1 the distances become small and decrease very slowly, therefore the minimum correct embedding dimension is 2. In this example $5 \times 10^3$ samples were used although $10^3$ samples are enough for reliable estimation. The results are equivalent for different $r$. Adding more than 4% noise to the series makes the estimation uncertain.

The number of data points should be large enough to unambiguously define the mapping $g$ and provide adequate density of points in phase space for finding at least a few close neighbour of every point. An insufficient number of data points may lead to the underestimation of the minimum embedding dimension because one may miss the parts of the phase space where the distances $d_{ij}$ are large.

![Figure 2.6](image)

**Figure 2.6** The distribution of points representing the distances is the same for $r = 0.2\%$ and $0.01\%$. The embedding dimension is 1; data are generated with the Hénon map, $N = 5 \times 10^3$, no noise added.
To decrease the influence of mapping slope $r$ has to be as small as possible while remaining large enough for finding enough neighbours. Its selection is a tradeoff between this two conditions. When the mapping has steep slopes, too big an $r$ will cause the over-estimating of the minimum dimension. Measurement noise and inadequate density of points in phase space limit the lower bound of $r$. To check the consistency of the estimation, the distances should be computed for several values of $r$. When the embedding dimension is smaller then the minimum correct one, the distribution of points in graphs will be the same for different $r$ (Fig. 2.6). When the dimension is equal or large then the minimum one, with decreasing of $r$ the distances will tend to small values.

Next example, Fig. 2.7, is the estimation of the embedding dimension for the Rössler system (2.7). The observable is the $x$ variable with 1% uniform noise added. The sampling time was $\tau = 0.5$. More data points (2 * $10^4$) are needed for this system than in the previous example because of its higher dimensionality. Minimum embedding dimension is 3. Note that for the dimension 2, the distances $d_{i,j}$ are large only in the small part of the phase space. As noted above, some form of averaging the distances over the phase space could lead to a wrong estimate. The estimation is uncertain after more than 2% noise added. Selection of a long sampling time result in a mapping with steep slopes. When the sampling time is large then 1.0 and 1% noise is added, in order to estimate if the minimum dimension is 3 or 4 one has to note that the distances are decreasing with decreasing of $r$ for $m = 4$.

When the state vectors of a system continuous in time are reconstructed by the method of delays, the delay time has to be estimated. Fraser's [1989a] proposal is to take time for which the mutual information between the components of reconstructed vectors is minimum as the delay time. We estimated the embedding dimension for several values of delay time. The delay time should be large enough so that the reconstructed two-dimensional trajectory does not lie along the diagonal line. With the increase of the delay
Figure 2.7 The distances between the images of close points for the Rössler system. The sampling time is 0.5. The scale for vertical axis is two times that of the horizontal axis. Noise of 1% amplitude is added, $r = 0.2\%$ and $N = 2 \times 10^4$. 
time the mapping between successive state vectors may become wild. When the aim of the reconstruction is building a predictive model of the observed system, the delay time should not be too long so that the class of mapping used for the approximation can approximate the given map with a small error.

The Mackey-Glass system [Mackey and Glass, 1977; Farmer, 1982b] is the example of an infinite dimensional system whose flow contracts onto sets of lower dimensions in the course of evolution

\[
\begin{align*}
\dot{x}(t) &= \frac{ax(t - \tau)}{1 + |x(t - \tau)|^10} - bx(t), \\
\end{align*}
\]

(2.23)

where \( a = 0.2 \), and \( b = 0.1 \). We integrated this system using the approximation proposed by Grassberger and Procaccia [1983]. With the delay of the system (not the delay of the delay method of reconstruction) \( \tau = 17 \) a small predictive error is achieved for the embedding dimension 4 [Farmer and Sidorowich, 1988]. The distances for this system are represented in Fig. 2.8. The sampling time is 6 and the value of \( r \) is 0.5%. The number of points is \( 5 \times 10^4 \). The distances for \( m \geq 3 \) do not decrease with decreasing of \( r \).

Last example is the estimating of the embedding dimension of Ikeda map [Ikeda, 1980] for \( \eta = 0.83 \)

\[
\begin{align*}
\dot{x}_n + 1 &= 1 + \eta \left( x_n \cos(u) - y_n \sin(u) \right), \\
\dot{y}_n + 1 &= \eta \left( x_n \sin(u) + y_n \cos(u) \right), \\
u &= 0.4 - 6.0 / \left( 1 + x_n^2 + y_n^2 \right).
\end{align*}
\]

(2.24)
Figure 2.8 The distances between the images of close points for the Mackay-Glass system with the delay 17. The sampling time is 6, $N = 5 \cdot 10^4$, $r = 0.5\%$, 1% noise is added. The scale for vertical axis is two times that of the horizontal axis.
Figure 2.9 The distances between the images of close points for the Ikeda map. \( N = 6 \times 10^4, r = 0.1\%, 1\% \) noise is added. The scale for vertical axis is two times that of the horizontal axis.
This map is not a diffeomorphism so the Takens theorem does not guarantee that the relationship between the true and the reconstructed state space is an embedding. The state space is reconstructed using \( x \) time series with 1\% noise added. The minimum embedding dimension is 4 (Fig. 2.9), twice higher than the number of degrees of freedom of the ‘observed’ system. This result is in agreement with the result of predictive method.

2.5 Relationship of the proposed approach to some previous ones

Our approach is similar to some previous approaches. The criterion we are using is equivalent to the criterion proposed by Packard et al., [1980]. Using the small width of conditional distribution as the criterion for estimating the embedding dimension corresponds to the ergodic theory approach to analysing a dynamical system. The criterion described with the equations (2.12) and (2.15) corresponds to differential topology approach. Of course, two approaches overlap. The selection of a particular criterion results in different procedures for the estimation.

Cremers and Hübler [1987] also check if a mapping is single-valued. The mapping they check assigns a flow vector to every point of phase space. Estimating flow vectors may be a problem when data are noisy. The way they check whether the mapping is single-valued is different from our way. Certainly, it would be better to have only one graph with several curves which indicate the correct embedding dimension but as we mentioned above even when the data are not noisy such a procedure may fail to produce a correct result. We rather rely on the human capability to perceive a form even in distorted images.
The illustrations in Fig. 2.3 and Fig. 2.4 demonstrate both why our criterion works and why the small predictive error indicates the correct estimating dimension. Conventional methods for approximating nonlinear mappings are applicable if the mapping to be approximated is single-valued. Although both criteria are related, a small predictive error is a stronger criterion. Another good feature of the small predictive error as a criterion is that this approach does not require a lot of data. Our approach needs two close points to check a given point of phase space. Unfortunately, building a predictive model is a computationally expensive task. Actually, we developed the proposed method in order to save time in building a predictive model.
chapter 3

Noise reduction in chaotic systems

On a longer time scale the behaviour of chaotic systems is practically unpredictable although their evolution is deterministic. The cause of unpredictability is the system’s sensitivity to initial conditions. Even the smallest error in determining the initial condition will be exponentially amplified in the course of prediction. But while the sensitivity to initial conditions puts limits to prediction, it also provides an opportunity for a substantial reduction of measurement noise.

3.1 Chaos provides information

In this chapter we will show how to determine from a sequence of imprecise measurement a state of chaotic system with much better precision than would have been possible from a single measurement. We will consider determining accurately only one state on the orbit, but the generalisation to noise reduction of a sequence is straightforward. It is assumed that the map of the system is known. When the map was known exactly and could be computed, we were able to reach an arbitrary accuracy in determining the states for several one- and two-dimensional chaotic systems we tested. When the map was not known exactly, the accuracy was determined by the accuracy in knowledge of the map.
Our method belongs to a class of optimisation algorithms. We are choosing the states around the observed state, which we want to determine with higher accuracy. The chosen state is the initial state of a sequence computed by iterating the known map. We are searching for the state for which the computed sequence is as close as possible to the observed sequence. Such a state is much closer to the exact state than the observed one.

We will explain intuitively why it is possible to have substantial reduction of noise in chaotic systems in section 3.1. In section 3.2 we will show how to measure the information about the chaotic state obtained by a sequence of measurements. The noise reduction method of Farmer and Sidorowich [1988] and Kostelich and Yorke [1990] will be described in section 3.3. We will explain our method and demonstrate it using the sequence generated by the logistic map in section 3.4. In sections 3.5 and 3.6 we will show how to determine states of invertible and noninvertible two-dimensional maps. The effect of the level of noise and the accuracy in knowing the map will be considered in section 3.7. Further improvements of the method will be discussed in the section 3.8.

3.1 Chaos ‘creates’ information

Any two initially close trajectories of a chaotic system eventually diverge from one another. On average, the distance between close trajectories grows exponentially for a short time

$$\varepsilon_t = \varepsilon_0 e^{\lambda t},$$

(3.1)

where $\lambda$ is the largest positive Liapunov exponent [Eckmann and Ruelle, 1985], $\varepsilon_0$ is the initial distance, and $\varepsilon_t$ is the distance after time $t$. For example, two initially close
trajectories of the Mackey-Glass system (2.23) for the delay $\tau = 17$ are represented in Fig. 3.1.

![Figure 3.1](image)

**Figure 3.1** The divergence of two initially close trajectories of the Mackey-Glass system; $\tau = 17$ (2.23).

After several iterations two trajectories are different. Suppose that we are observing the system whose evolution is governed by some smooth map. Measurements are taken with limited resolution. Taking one measurement as the initial state we start computing new states, that is, we start predicting. After several iterations the difference between the predicted and measured states will be bigger than the measurement resolution.

Only several steps after the measurement we 'lost' information about the evolution of observed system. If we perform a new measurement we will be 'surprised' to find the system in the state different from the one we predicted, that is, we will obtain new information. To know where in the phase space the chaotic system is we have to monitor the system continuously\(^1\).

\(^1\) Nice description of relationship between chaos and information is given in Shaw, [1981].
Figure 3.2  The system is somewhere in the interval $a_k$ at time $t - \tau$. When the possible trajectories of the system are known, and if the system is somewhere in the interval $a_j$ at the time $t$, then the interval where the system may be at time $t - \tau$ is reduced to $a_j^{-1}$.

With each new measurement we have more information about the position of the system in the past [Farmer, 1982]. We illustrated this in Fig. 3.2. At the times $t - \tau$ and $t$ the system is somewhere inside the intervals $a_k$ and $a_j$ respectively. The width of intervals is determined by the measurement resolution. Knowing possible trajectories of the system we know that the system was at time $t - \tau$ somewhere inside the interval $a_j^{-1}$.

3.2 Measuring the information acquisition rate

Suppose we observe the dynamical system described as a mapping $f$

$$f : X \rightarrow X,$$  \hspace{1cm} (3.2)

where $X$ is the phase space of the system. The measurement is performed with finite precision so we see the phase space $X$ of the system as partitioned into finite areas $A_i$. The diameter of the elements of partition $A_i$ correspond to the available measurement precision. We will assume that the $A_i$ are pairwise disjoint and cover the whole phase
space $X$. All points $x_i \in X$ from the phase space falling in the same partition $A_i$ will be indistinguishable. The set of elements $A_i$ will be denoted as partition $\alpha$

$$\alpha = \{ A_1, A_2, \ldots, A_N \}. \quad (3.3)$$

The information content of a partition $\alpha$ is defined [Peterson, 1983] as

$$H(\alpha, f) = -\sum_{i=1}^{n} \mu(A_i) \log_2 \mu(A_i), \quad (3.4)$$

where $\mu(A_i)$ is the probability to find the state of the system in the partition $A_i$. The information content is the measure of the amount of uncertainty of knowledge where the system is after the observation.

The trajectory of the system is seen as a sequence of the elements of partitions

$$x_0, x_1, \ldots, x_k$$

$$\downarrow$$

$$a_0, a_1, \ldots, a_k,$$

where $a_i \in \alpha$ and $x_i \in a_i$. Observing imprecisely the trajectory of the system $\{a_i\}$, $i = 0, \ldots, k$, we know that at time 0 the system must have been somewhere inside the intersections

$$x_0 \in a_0 \cap f^{-1}(a_1) \cap \ldots \cap f^{-k}(a_k), \quad (3.6)$$

where $f^{-i}$ is the i-th backward iterate of the map $f$. A sequence of several imprecise measurements allows determination of states with better
precision than the partition $\alpha$. The sequence of $k$ observations induces a finer partition

$$\alpha_k = \alpha_0 \land f^{-1} \alpha_1 \land \ldots \land f^{-k+1} \alpha_{k-1},$$

(3.7)

that is the least common refinement of the partition $\alpha_0, f^{-1} \alpha_1, \ldots, f^{-k+1} \alpha_{k-1}$. The limit

$$h(\alpha, f) = \lim_{n \to \infty} [H(\alpha_{n+1}) - H(\alpha_n)] = \lim_{n \to \infty} n^{-1} H(\alpha_n),$$

(3.8)

gives the rate of information creation with respect to the partition $\alpha$. The maximum information rate when the partition is varied is called Kolmogorov-Sinai or metric entropy

$$h(\alpha) = \sup_{\alpha} h(\alpha, f).$$

(3.9)

It can be thought of as the measure of the rate of information about the state $x_0$ gained by each new observation. The partition $\alpha$ is called a generator if any state of the system can be determined to any desired degree of accuracy in terms of some finite-length compound of $\alpha$. Kolmogorov defined the entropy of dynamical system $f$ to be $h(\alpha, f)$ if $f$ has a generator $\alpha$ or $\infty$ otherwise. It is proved that Kolmogorov's definition and the definition (3.9) are equivalent [Peterson, 1983].

### 3.3 Previous approaches to noise reduction

The metric entropy of a chaotic system is larger than zero. The sequence of imprecise measurements can be used for the computation of a sequence closer to the true states than the observed ones. We will denote the observed states as $o x_t$, true states as $x_t$, and

---

2 This definition is given by Sinai [Peterson, 1983].
the computed states as $\mathbf{e}x_t$. All states are $m$-dimensional vectors. We assume that the observed time series $\{\mathbf{o}x_t\}$ is in the form

$$\mathbf{o}x_t = \mathbf{x}_t + \xi_t,$$

where

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t),$$

and elements $\xi_t$ characterise measurement noise. The elements $\xi_t$ are uncorrelated.

The task of the noise reduction is to estimate states $\mathbf{e}x_t$ that are closer to the true states $\mathbf{x}_t$ than the observed states $\mathbf{o}x_t$. We will assume that the observed sequence is long enough to reach the desired level of noise reduction.

The method for noise reduction proposed by Farmer and Sidorowich [1988] is based on the idea of mapping the successive measurements to the same point in time and averaging them together. The nonlinear map of the system is known 3. Assuming that the noise is small compared to the nonlinearities of $f$, so that the dynamics is locally linear, they derived the equation

$$\mathbf{c}x_t = \left( \sum_{j=-\beta}^{-1} \Theta_j \right)^{-1} \sum_{j=-\beta}^{-1} \Theta_j f^j(\mathbf{o}x_{t,j}),$$

where

$$\Theta_j = \left( \left[ df^i(x_{t,j}) \right]^T df^i(x_{t,j}) \right)^{-1},$$

where $df^i$ is the first derivative if the $i$-th iterate of the map $f$. $\beta$ measurements after the time $t$ and $\nu$ measurements before the time $t$ are mapped to the time $t$ and averaged with the weighting factors contained in the $m \times m$ symmetric matrix $\Theta_j$. The weighting factors depend on local expansion and contraction rates. The equation (3.12) can be

3 If unknown system is approached, the first step would be estimation of its map.
applied recursively. In the examples where the map of a chaotic system is known exactly, with this method the noise can be reduced for more than ten orders of magnitude.

Another method for noise reduction is proposed by Kostelich and Yorke [1988, 1990]. Like Farmer and Sidorowich they assumed that the map \( f \) is locally linear. Their first step is to find a linear approximation in the small region of the phase space

\[
0x_{t+1} = L(ox_t) = Ax_t + b
\]  

(3.13)

where \( A \) is a \( m \times m \) matrix, and \( b \) is \( m \)-dimensional vector. The computed sequence \( \{ cx_t \} \) corresponding to the observed trajectory in the given small region of phase space is the minimum of the functional \( O \)

\[
O = \sum_t \left\{ \| cx_t - ox_t \|^2 + \| cx_t - L(cx_t) \|^2 + \| cx_{t+1} - L(cx_t) \|^2 \right\}
\]

(3.14)

where the sum runs over all points along the observed trajectory. The computed points are simultaneously closest to the measured ones and best satisfy the corresponding local linear approximation of the map. The numerical experiments suggest that this method can reduce noise levels up to a factor 10 of the initial level of the noise.

3.4 Determining a state with improved precision by maximising the duration of correct predictions

The noise reduction achieved by Farmer and Sidorowich is quite substantial. Stimulated by their result we investigated how to achieve an arbitrary level of noise reduction when the map is known exactly. We considered reducing the measurement noise of only one state, that is, determining one state with better precision. Beside the assumption that the
noise is uncorrelated we also assumed that the measurement noise is bounded

$$\| \xi_t \| < \delta .$$  (3.15)

We used the same metric as in the research on estimating the embedding dimension

$$\| x_t \| = \max \{ | x_{t,i} | \} ,$$  (3.16)

$$1 \leq i \leq m$$

where $m$ is the dimension of the system and $x_{t,i}$ is the component of the state $x_t$. This norm was used for computational efficiency.

Suppose that we want to determine with better accuracy the state observed as $ox_t$, $1 < t < N$. We know that the exact state $x_t$ is somewhere in the interval $ox_t \pm \delta$ and we know the map of the system. If we choose an arbitrary state $cx_0$ from this interval and start computing new states of the system $cx_i$, the distance between the computed and observed state will be larger than the maximal amplitude of noise $\delta$ after approximately $k$ iterations

$$k > \lambda^{-1} \ln( \delta / \| x_t - cx_0 \| ) .$$  (3.17)

The closer the chosen state is to the exact state the longer will the computed states stay close to the observed states. Our approach to noise reduction is posed as an optimisation procedure like Kostelich and Yorke approach. The function we are maximising is

$$L = \Phi( cx_0 ) ,$$  (3.18)

the length of the computed sequence in which all elements are closer than $\delta$ to the
observed states,

$$\| o x_{t+i} - c x_i \| < \delta \quad \text{for all} \quad i = 1, \ldots, L .$$

(3.19)

We search for the initial state $c x_0$ for which the length $L$ is maximal. The value of $c x_0$ for which the objective function (3.18) has a maximum is much closer to the exact state than the observed one. Our optimisation algorithm is recursive. At the beginning of the algorithm, the initial states $c x_0$ are chosen randomly in the interval $o x_i \pm \delta$ and the length $L$ is computed. After $T$ trials the smaller area in the phase space, where the maximum of objective function may be, is localised. This area is searched for the maximum at the next level of recursion. At each level of recursion the search area is decreased, that is the desired state is determined with better precision. The algorithm halts when the length $L$ stops decreasing.

Let us illustrate this approach by trying to determine a position on the orbit of the logistic map (2.13) with better precision. The logistic map was iterated starting with the initial state $x_0 = 0.82$. The control parameter is $\eta = 3.9$. Uniform noise with the amplitude $\pm 1\%$ of the maximum amplitude of true states is added to the states $x_t$ ($\delta = 1\%$) to simulate the measurement noise. The goal is to determine the state $o x_0$ with better accuracy. At the first level of recursion the initial states $c x_0$ are chosen randomly in the interval $o x_0 \pm \delta$. After $T = 2000$ trials the initial condition $o^1$ for which the computed series was longest is selected. The length of the computed sequence whose all elements are closer then $\delta$ to the observed states is plotted against the initial state in Fig. 3.3a. At the next level of recursion the search for the maximum is repeated in the interval $o^1 \pm \delta / S$ (Fig. 3.3b), where the constant $S$ is larger than one ($S = 10$). At the 16-th level of recursion the machine accuracy is reached (Fig. 3.3c). The machine accuracy is about 18 orders of magnitude smaller than the noise and we needed 80 imprecise ‘measurements’ for reaching it. At the k-th level of recursion the interval of the search is $o^{k-1} \pm \delta^{k-1} / S$, where $o^{k-1}$ and $\delta^{k-1}$ are the location of the maximum and the half-size of the interval of the previous level of recursion respectively.
Figure 3.3 The length $L$ is plotted against the initial states $x_0$. The 'observed' sequence is generated by iterating the logistic map, $\eta = 3.9$, and adding uniform noise $\delta = 1\%$. The true state to be determined is 0.82. The observed sequence consists of 200 elements. a) At the first level of recursion the maximum is localised after 2000 trials. The region for the search at the next level is shaded. b) The maximum at the next level. c) The objective function after the machine precision is reached.
Figure 3.4 Error caused by sparse sampling. The parameters are the same as at Fig. 3.3 except the true state is 0.61, at the third level of recursion, the algorithm chooses the wrong region around the right peak for the next search.

Figure 3.5 The logarithm of distances between the orbit of the logistic map starting at $x_0 = 0.61$ and orbits starting at $x = x_0 + i \, dx$, is plotted against $i$ for the first 31 steps; $dx = 1.4 \times 10^{-7}$.

The number of trials $T$ should be big enough to avoid trapping in local maxima caused by sparse sampling. For instance, in Fig. 3.4 when the true state is 0.61 and all other parameters are the same as in the first example, at the third level of recursion the
algorithm chooses a wrong region for the next search (around the right peak in Fig. 3.4). The problem does not appear when the number of trials is doubled. To see why the spurious maxima can be found we draw\(^4\) in Fig. 3.5 the logarithm of distances between the orbit starting at \(x_0 = 0.61\) and orbits starting at \(x = x_0 + i \ dx\), where \(dx = 1.4 \times 10^{-7}\), in the first 31 steps. In the case of sparse sampling the location marked by the arrow can be selected as the position of maximum.

### 3.5 Two-dimensional invertible mapping

In the case of the logistic map the objective function (3.18) is well behaved that is, local maxima do not cause difficulties during the optimisation. In the case of two-dimensional maps the objective function has many local maxima. Localising the area for search at the next level of recursion solely on the basis of maximum of (3.18) takes prohibitively large number of trials \(T\). For example, the structure of the objective function (3.18) for the Hénon map (2.14) is shown at Fig. 3.6. The true state is in the middle of the square. The initial states are chosen randomly in the interval \((0.935... \pm 1.4, 0.132... \pm 0.45)\). Fig. 3.6 shows that the peaks of the objective function are localised along the line.

The true states must be somewhere inside the area where peaks are. Note that the line defined by the peaks in Fig. 3.6 pass near the true state. We will define the set \(C_f\) as a set of initial states for which the length \(L\) of the computed series is larger than some number \(L_f\),

\[
C_f = \{ \ cx_0 \ | \ \| ox_{i+k} - cx_k \| < \delta \ \text{for all} \ k, \ 0 < k \leq L, \ L > L_f \} ,
\]

where \(cx_0\) are the points in the area around \(ox_i\).

\(^4\) The graph has a fractal structure. Drawing it at different resolution reveals similar structure.
Figure 3.6  The places where the value $L$ of the objective function (3.18) for the Hénon map is larger than the chosen number. The large square is the area determined by the measurement noise. The true state is in the middle of the smaller square.

The Hénon map is invertible, and it's inversion is

$$x_n = \frac{1}{0.3} y_{n+1}, \quad (3.21)$$

$$y_n = x_{n+1} - 1 + \frac{1.4}{0.3^2} y_{n+1}^2.$$
In the same way we used the forward Hénon map to find the states closer to the true states than the observed ones, we can use the backward Hénon map (3.21). Again we can randomly choose the initial states \( c x_0 \) around the \( o x_i \) and define a set \( C^b \)

\[
C^b = \{ \ c x_0 \ | \ \| o x_{i+k} - c x_k \| < \delta \ \text{for all} \ k, \ 0 < k \leq L, L > L_b \} . \quad (3.22)
\]

The elements of the set \( C^b \) are the initial states for which more then first \( L_b \) states computed with the backward map are closer then \( \delta \) to the observed states. For large \( L_b \) this set can be represented as a curve as well. In the Fig. 3.7 we show the sets \( C_f \) and \( C^b \) for the Hénon map. The curves corresponding to these sets are crossing or touching in the vicinity of the true state.

The set \( C_f \) is approximately the local stable manifold \( V^s(x_i, \varepsilon, \lambda) \) at the \( x_i \), in the area of search. The local stable manifold is a set of points in phase space defined as [Eckmann and Ruelle, 1985]

\[
V^s(x, \varepsilon, \lambda) = \{ \ y \ | \ d(f^t x , f^t y ) \leq \varepsilon e^{\lambda t} \ \text{for all} \ t \geq 0 \} , \quad (3.23)
\]

where \( d \) is the Euclidian distance, \( \varepsilon > 0 \), and \( \lambda < 0 \). The set \( C^b \) is approximately the local unstable manifold \( V^u(x_i, \varepsilon, \lambda) \) defined as

\[
V^u(x, \varepsilon, \lambda) = \{ \ y \ | \ d(f^{-t} x , f^{-t} y ) \leq \varepsilon e^{\lambda t} \ \text{for all} \ t \geq 0 \} . \quad (3.24)
\]

Stable and unstable manifolds\(^5\) for the Hénon map are represented in Fig. 3.8. For this map the unstable manifold is indistinguishable from the Hénon attractor. The local stable and unstable manifolds of the Hénon map have the form of curves. The stable and unstable manifolds of other two-dimensional diffeomorphic dynamical systems have a similar geometrical description [Eckmann and Ruelle, 1985].

\(^5\) Stable and unstable manifolds are unions of local manifolds, \( V^i = \cup_{t > 0} f^{-t} V^i(x, \varepsilon, \lambda) \), where \( i \) is s or u.
Figure 3.7 The sets $C^f$ and $C^b$ for the Hénon map (2.14) when the state to be determined is $(x = 0.9354122..., y = 0.132646...)$.
Uniform noise $\pm 1\%$ is added to each component of the state. a) At the first level of recursion $L_f = 12$ and $L_b = 2$. The true state is in the middle of square. b) At next level $L_f = 14$, and $L_b = 4$. 
The curves determined by the elements of sets $C_f$ and $C_b$ can be found at each level of recursion. In most cases the curves can be represented as straight lines. When the lines have almost the same slope, it is necessary to use the quadratic representation\(^6\). To determine with better precision a state of invertible map we find\(^7\), at each level of recursion, the minimum lengths of forward $L_f$ and backward sequence $L_b$ such that the corresponding sets of initial states $C_f$ and $C_b$ can be represented as curves. The location of the intersection of the curves is the centre of search at the next level of recursion. At each level, the region for the search is decreased. In Fig. 3.7b are shown sets $C_f$ and $C_b$ at the second level of recursion. To reach the machine precision in this example we needed 20 states before and 80 states after the desired states (the machine accuracy in this case is about 13 orders of magnitude smaller than the maximum amplitude of noise).

---

\(^6\) In the case of Hénon map, the fitting of curves with straight lines was enough. When the lines have almost the same slope we decreased area for search at next level of recursion more slowly.

\(^7\) In our implementation of the algorithm this part is done interactively.
3.6 Two-dimensional noninvertible map

When the chaotic map is not invertible, the set of points $C^b$ is not defined. In that case a useful set for localising the area where the exact state may be is defined as

$$\text{Cd} = \{ cx_0 \mid \| ox_{i+k} - cx_k \| < \delta \text{ for all } k, \ 0 < k \leq L, \ L > L_d \} . \quad (3.25)$$

To find the elements of this set we search for the initial states randomly chosen around the state observed $B$ steps before the desired state, for which more then first $L_d$ states computed with the forward map are closer then $\delta$ to the observed states. For found initial states we take the states computed after $B$ steps as elements of the set $\text{Cd}$. In some cases this set can be represented as a curve. In Fig. 3.9 are represented sets $\text{Cf}$ and $\text{Cd}$ for the Ikeda\(^8\) map (2.24) at the first level of recursion.

When the sets $\text{Cf}$ and $\text{Cd}$ can be described as the curves, again the area for the search at the next level of recursion is around the place of crossing or touching of the two curves. The novelty is now, that at each level of recursion we determine with better precision not only a desired state but several previous states as well. For example, at the second level of recursion the initial states $cx_B$ are chosen randomly in the area localised at the first level of recursion around the observed state $ox_B$. If we have $n$ levels of recursion, it is necessary to determine with better precision the states $ox_{i-(n-1)B}, \ldots, ox_{i-B}, ox_i$.

---

8 Ikeda map is invertible. In this example we 'pretend' that it is not invertible.
The sets $C_f$ and $C_d$ for the Ikeda map (2.24) when the state to be determined is $(x = 0.61205..., y = -0.84571...)$. Uniform noise $\delta = 1\%$ is added to each component of the state. At the first level of recursion $L_f = 7$ and 20000 trials are used to find $C_f$. To find $C_d$ 10000 trials are used, $L_d = 6$.

The sets $C_f$ and $C_d$ cannot be represented as curves for every two-dimensional map. For instance, in Fig. 3.10 are shown the sets $C_f$ and $C_d$ for the Ushiki map [Eckmann and Ruelle, 1985]

$$x_{n+1} = (3.7 - x_n - 0.15 y_n) x_n,$$
$$y_{n+1} = (3.7 - 0.15 x_n - y_n) y_n.$$ (3.26)

In such cases we are partitioning the phase space and at each level of recursion we are searching for the partitions where elements of both set $C_f$ and $C_d$ are. This version of the algorithm is most general but the price for its application to maps like the Hénon map is higher computational effort.
Figure 3.10 The sets $C^f$ and $C^d$ for the Ushiki map (3.26) when the state to be determined is $(x = 1.577..., \ y = 1.2648... )$. Uniform noise $\pm 1\%$ is added to each component of the state.  

a) The first level of recursion $L_f = 4$ and $L_d = 5$. 

b) The first level $L_f = 5$, and $L_d = 6$. 
3.7 The influence of the strength of noise and uncertainty in knowing the map

Our numerical simulations suggest that strong measurement noise does not cause any problem in the described approach. For example, in the Fig. 3.11 are shown sets $C^f$ and $C^b$ in the case of the Hénon map with 20% noise added. The other parameters are the same as in Fig. 3.7. It can be seen that curves cannot be represented as lines, but the place of the crossing of curves is around the true states.

![Figure 3.11](image)

**Figure 3.11** The sets $C^f$ and $C^b$ for the Hénon map at the first level of recursion. All parameters are the same as in the Fig. 3.7 except that 20% noise is added; $L_f = 12$ and $L_b = 2$.

The accuracy in determining the desired state is limited by the uncertainty in knowing the chaotic map. When the map is not known exactly, the uncertainty in determining the
state we achieved is of the same order of magnitude as the uncertainty in knowing the map. We define the uncertainty $\chi$ in knowing the map as

$$\chi = \sup_{x_i \in M} \left\{ |x_i(t+1) - f^*(x_i(t))| \right\},$$

(3.27)

where $f^*$ is the approximation of the map, and $M$ is the phase space of the chaotic system.

### 3.8 Further improvements and discussion

We considered the determination with higher accuracy of only one position on the chaotic orbit, but the construction of the algorithm for better determination of the part of the orbit (or noise reduction) would be straightforward. The goal of our research was to investigate the methods for reaching an arbitrary accuracy in the cases where it is, in principle, possible. The constructed algorithm is not as efficient as Farmer and Sidorowich or Kostelich and Yorke method. The use of our method is recommended for the cases where high accuracy is necessary.

The efficiency of the method can be improved. At each level of recursion we were randomly probing the objective function. This approach was selected because of its algorithmic simplicity. With the increase of the dimensionality of the system phase space the number of necessary trials is growing exponentially. In all cases we tested the objective function has a topology that would allow more efficient search strategies. For example, one approach could be a group of 'hill climbers' starting a search from random initial states. Also, it would be simple to parallelise our method, and that would be another way to improve its computing efficiency.
Discovering heuristics for constructing neural networks

The study of neural networks started as a modeling of a behavior of group of interconnected neuronal cells. Formally, networks are a class of mappings so they can be used for constructing nonlinear models. For example, Lapedes and Farber [1987] used them for predicting points in chaotic time series and approximating nonlinear transfer functions. They argue that good features of networks are their abilities to provide explicit, analytic, global approximation and have very small data requirements. Unfortunately, with increasing the complexity of the mapping to be approximated it is harder and harder to determine the parameters of the network, and, in some cases, it may be practically impossible. In this chapter we will describe our approach to this problem using a particular example.

We were constructing deterministic feed-forward neural networks by the backpropagation algorithm in order to learn more about the processing in the visual cortex. The aim was to develop a network with one hidden layer whose units in the output layer
act as detectors of the velocity of one-dimensional patterns. With the standard algorithm we were able to determine the parameters of the network only for networks with one or two neurons in the output layer. Intuitively, one can expect that the knowledge obtained studying similar but simple systems may be useful for the construction of the larger system. We analysed successful network's configurations, and with the obtained knowledge we were able to construct larger networks. The solution is similar to the artificial intelligence approach to solving problems: to solve a hard problem try to first solve a similar simple one.

In the first three sections will be described what is deterministic feed-forward neural network and how to determine its parameters. Also, we will mention the problems that can arise in the process of determining the parameters as well as some approaches to overcome them. In section 4.4 we will show the analogy between the construction of neural networks and problem solving in artificial intelligence. In following two section we will describe the network we wanted to construct and analyse the successful configurations. The procedure for constructing a larger network is described in section 4.7.

4.1 Feed-forward neural network

In neural networks, a given mapping is approximated by using as a set of basic functions a set of sigmoid functions $\phi(z)$

$$y = \phi(\sum_i w_i x_i - \theta).$$

(4.1)

The equation (4.1) is one of the simplest model of a neuron. In a first versions of this model [McCulloch and Pitts, 1943] the output $y$ of the neuron and the inputs $x_i$ can take the binary values of 0 and 1. The neuron fires if the linear combinations of its
inputs $\sum_i w_i x_i$ is larger than the threshold level $\theta$. The coefficients $w_i$ are connection strengths that characterise the interaction between two neurons. In the continuous version of this model [Hopfield, 1984] output and inputs can take any value between 0 and 1. These values may be thought of as short-time firing rates of neurons. The connection strengths are positive or negative real numbers. The function $\phi(z)$ is a differentiable sigmoid function. In our research we used the function

$$\phi(z) = \frac{1 + \tanh(z)}{2} = \frac{1}{1 + \exp(-2z)}.$$  

(4.2)

A deterministic feed-forward neural network consists of several layers of neurons. Each neuron in a layer receives inputs from all neurons in the lower layer and sends output to all neurons in the upper layer. In the case of the network with one hidden layer, a composition of sigmoid function $f(x, a)$ has the form

$$f : X \rightarrow H \rightarrow Y.$$  

(4.3)

The input $x \in X$ is first mapped onto the hidden (or internal) space $H$ and then from the hidden space to the output $y \in Y$. The output $h_i$ of neurons in the hidden layer is

$$h_i = \phi(z_i) = \phi(\sum_j w_{ij} x_j - \theta_i),$$  

(4.4)

where $x_j$ are components of the input. The output $y_k$ of neurons in the output layer is

$$y_k = \phi(z_k) = \phi(\sum_i w_{ki} h_i - \theta_k).$$  

(4.5)

Recently, Funahashi [1989] proved that any continuous mapping can be approximated by the composition (4.3).
4.2 Determining the parameters of the network

The parameters of a neural network, that is the connection strengths and thresholds levels can be determined by the ‘back-propagation’ algorithm [Werbos, 1988; Rumelhart et al, 1986]. The mapping to be approximated, that is the ‘true mapping’, is given as a set of input-output pairs \{x_p, y_p\}. The back-propagation algorithm is a search in parameter space for the parameters of an approximation that is close to the true mapping. The quality of approximation is characterised by the mean square error between the desired output and actual output of the network

\[ E_i = \sum_p \sum_j (y_{pj} - f_j(x_p, a_i))^2, \]  \hspace{1cm} (4.6)

where \(y_{pj}\) is the j-th component of the desired output \(y_p\), and \(f_j(x_p, a_i)\) is the output of j-th neuron in the output layer when the input is \(x_p\) and the parameters of the network are \(a_i\). The search in the parameter space is governed by the difference equation

\[ a_{i+1} = a_i - \eta \frac{\partial E_i}{\partial a}, \]  \hspace{1cm} (4.7)

where \(\eta\) is some constant. As it can be seen from the equation (4.7) the back-propagation algorithm is essentially the gradient descent method for the optimization. The use of the conjugate gradient method [Lapedes and Farber, 1987] does not qualitatively change the process. It only provides faster search in some cases. The distinctive feature of back-propagation is the chain rule\(^1\) for computing the gradient

\[^1\text{Note that the chain rule is nothing novel in mathematics.}\]
of (4.6). The partial derivatives of the functional (4.6) are

for output neurons

\[ \frac{\partial E}{\partial \theta_k} = \sum_p (y_{p,k} - \phi(z_k)) \phi'(z_k) , \]  

(4.8a)

\[ \frac{\partial E}{\partial w_{ki}} = \sum_p (y_{p,k} - \phi(z_k)) \phi'(z_k) h_i , \]  

(4.8b)

for hidden neurons

\[ \frac{\partial E}{\partial \theta_i} = \sum_p [\phi'(z_i) \sum_k w_{ik} (y_{p,i} - \phi(z_k))] , \]  

(4.8c)

\[ \frac{\partial E}{\partial w_{ij}} = \sum_p [x_j \phi'(z_i) \sum_k w_{ik} (y_{p,i} - \phi(z_k))] . \]  

(4.8d)

Although the back-propagation is posed as an optimization algorithm it is not necessary to find the global minima of the functional (4.6). It is enough to find the parameters for which the local minimum is deep enough, that is \( E \leq \rho \), where \( \rho \) is the maximum acceptable level of the (4.6). This criterion is a rather weak requirement on the solution of the search in parameter space. When \( p \) and \( j \) are large, there may be many network configurations satisfying this requirement. This is one reason for the relatively good results of the rather unsophisticated search algorithm as the back-propagation. Even the configuration for which \( E \leq \rho \) should be taken cautiously. In some cases measuring proximity of two functions with (4.6) is not sufficient [Vapnik, 1982]. The network may approximate the desired mapping sufficiently well almost everywhere except in a small area in domain where a large deviation may happen. Fig. 4.1 shows an example of such a case taken from Vapnik [1982]. To exclude such a situation, it is necessary that the network approximate the desired mapping uniformly over the whole domain of the
definition of the mapping. It would be better to use a stronger requirement for the ‘quality of the approximation’

\[ E_i = \sup \| y_{p,j} - f_j(x_p, a_i) \| , \]

but this functional would need a bigger computational effort.

![Figure 4.1](image)

**Figure 4.1** The approximation \( f(x, a) \) of the function \( g(x) \) would be characterized as a good one using mean-square distance between two functions. After Vapnik [1982].

### 4.3 Bad topology of the objective function and the problem of scale

When the search for a good configuration is governed by the equation (4.7) the solution is guarantied only if the objective function (4.6) is ‘well behaved’. Intuitively, the topology of the objective function should be such that there are only deep local minima corresponding to good approximations. Unfortunately, when the mapping to be implemented is complex, the objective function usually does not have the desired topology. One of the problems caused by spurious local minima may be solved by adding the noise of different amplitudes into the equation (4.7). The noise is dependent
on a parameter ("the temperature") which reduces to zero over time according to a cooling schedule. The method is called simulated annealing [Kirkpatrick et al., 1983]. The noise prevents parameters of the system from settling in shallow minima. The method is successful when spurious local minima are not deep. Another approach to the problem of bad topology would be to use a better behaved objective function or to try a different search strategy. Muhlenbein [1989] proposed the use of a genetic algorithm for searching in parameter space. Instead of having one search he has a 'population' of searches. After some number of iterations the uninteresting parts of parameter space are discarded and the searches continued in more promising regions. Teasuro and Sejnowski's [1989] approach to the problem is to transform the input and output space in such a way that the objective function is well behaved.

Even when the topology of the objective function does not pose any problems, there is still the problem of scaling. With the increase of the complexity of the mapping to be estimated, the size of the network may become prohibitively large [Minsky and Papert, 1988]. More parameters to be determined require much longer computational time. Moody's [1988] approach to both problems is not to use the sigmoids (4.1) as basic functions but to use more complex functions like B-splines with different scales. Because of the complexity of splines no hidden layer is needed. A hashing technique further reduces the computational effort. This approach gives good results when the mapping is sparsely defined over its domain. Brent [1989] and Waibel [1989] also offered interesting solutions2. Brent's approach is to construct first a suitable decision tree and then construct a neural net from it. Waibel's approach is first to split a mapping into parts and estimate the parts as smaller nets. These small nets are later 'glued' together to produce the desired mapping.

---

2 Both approaches studied mappings from real spaces onto binary spaces.
4.4 Heuristic search

One aspect of research in artificial intelligence (AI) is making computers solve problems, such as proving a theorem or solving a puzzle [Rich, 1983]. A possible approach is to view the solving of a problem as a heuristic search in the problem space. The easiest way to explain this approach is to consider a simple problem.

In the 8-Puzzle [Pearl, 1985] we are given a square $3 \times 3$ board on which are placed 8 numbered square tiles (Fig. 4.2). There is one empty square on the board. One of the tiles adjacent to the empty square can be slid into it. The objective is to transform the initial configuration of tiles into the goal one by sliding tiles. The states of the problem space of this puzzle are possible configurations of tiles. Solving the puzzle requires a search in the problem space for a goal state.

![Figure 4.2](image)

**Figure 4.2** The initial and goal configuration of the 8-Puzzle, and the first 3 move options.

Even in this simple problem moving randomly in the problem space is not practical. Interesting problem spaces have immense number of configurations. Like human problem-solving [Newell, 1980], AI is controlling the search by heuristics. Heuristics
are informal judgemental strategies or rules for choosing the direction of the movement in the problem space [Pearl, 1985]. The use of heuristic rules does not necessarily lead to a solution. Heuristics are educated guesses that usually work. In the 8-Puzzle a heuristic rule can be

- to reach the goal state move from the current state to the state for which the number of mismatched tiles between it and the goal state is minimum.

In the example in Fig. 4.2, the state A would be selected after the initial state according to the given heuristic rule. The simplest heuristics search strategy is ‘hill-climbing’. The next state is selected only according to the proximity to the goal state. The previous path through parameter space is not remembered. This strategy is very popular because it requires little computational effort. Note that it is equivalent to the search strategy used by the back-propagation algorithm. In AI to overcome the problems that can be encountered in the course of search more complicated strategies for a heuristic search are used, like depth-first, backtracking, breadth-first or hybrid strategies [Pearl, 1985]. These strategies can be used for constructing neural networks as well.

The search in back-propagation is governed by only one rule, proximity to the goal state. Having more rules or having a more detailed rule can help find the goal state. The problem is how to get useful heuristics. In AI they are discovered by analysing simplified models of the problem domain [Pearl, 1985]. There is also a more ambitious approach, where the heuristics are searched for in an automated way in analogous problems domain [Lenat, 1984]. Again, the search for useful heuristics is governed by heuristics.

Our approach to overcoming the problems we encountered in trying to construct a neural network is equivalent to the AI approach used to obtain and later apply relevant
heuristics. We analysed the successful configurations of neural networks trained to solve simple tasks. Analysing smaller neural nets we were able to:

- estimate the necessary numbers of neurons;
- estimate the useful ranges of the parameters of the network;
- discover potential functional dependencies between the parameters.

The obtained rules were used to guide the search in parameter space of large neural networks. We determined the parameters of the large networks again using back-propagation, but with the obtained knowledge we used a smaller number of components and knew where not to search for a good configuration. In our case the discovered functional dependencies among the parameters drastically reduced the size of the parameter space. For example we discovered a simple linear relationship between some parameters. Instead of searching across a two-dimensional subspace we had to search along the line $w_k = -w_i$ (Fig. 4.3).

![Figure 4.3 The reduction of the parameter space.](image)
4.5 Constructing a small network

Constructing an artificial neural network to perform a given task is a way to understand some aspects of biological neural systems. At this stage of technology it is possible to experimentally determine anatomical connections between neurons, but it is extremely hard to discover functional connections [Getting, 1989]. However, from behavioural and psychophysical experiments the task of some parts of the system can be discovered. The part of the system can be represented as neural network and its task can be represented as a functional relationship between the input and output layer. Knowing the anatomy of the network and the mapping the network implements one can use the algorithms like back-propagation to learn something more about the potential connections between the neurons.

We were investigating how neurons can be connected in order to extract the velocity of images. The motion of the image across the retina is caused by the combined motion of parts of observed scene and the motion of the observer. The information of the image motion is very important for biological systems [Gibson, 1979] and it has been hypothesized that it is the first evolutionary step in acquiring vision [Horridge, 1986]. Many neurons that respond strongly when stimulus moves with a particular velocity are found [for a review see Hildreth and Koch, 1987]. The inputs to the network we wanted to construct were one-dimensional patterns moving with the uniform velocity. They are thought of as the cut through an image. The intensity of pattern elements could take any value between 0 and 1. We wanted neurons in the output layer to perform as velocity detectors. In some cases their response was selected as 1 when the heterogeneous pattern moves with the appropriate velocity and 0 otherwise. In other cases their response was selected as a Gaussian function of pattern velocity with the peak at the appropriate velocity.
The architecture of the network is described as (4.3). The only additional element is that hidden neurons receive input both directly and after a delay $\tau$ (Fig. 4.4). This is quite plausible, since the delay of signals is common in biological systems. For example, neural delay lines were recently found in experiments on auditory discrimination of azimuth information in barn owls [Carr and Konishi, 1988]. It is believed that a mechanism involving a delay is responsible for velocity-sensitivity in ganglion cells of the rabbit retina [Barlow and Levick, 1965].

**Figure 4.4** The deterministic feed-forward neural network designed to extract velocity of moving images. Inputs reach the hidden layer both directly and after delay $\tau$. Units in the output layer are velocity detectors.

We tried to determine the parameters of the network by the back-propagation algorithm. The training set consisted of a few thousand patterns whose uniform velocity was restricted to integer values (the distance is measured in units of interreceptor spacing and
time in units of the delay time $\tau$). The distribution of the set of velocities of training patterns was uniform. During the search through parameter space, a part of the training set would be randomly selected. The gradient of the functional (4.6) was computed after sweeping through the selected part.

Interesting and reliable configurations were discovered when the training set consisted of random patterns and almost homogeneous (intensity of pattern elements was $\pm 5\%$ of the average intensity) patterns. When the structure of the patterns is random, the only information about motion is contained in the displacement of the pattern. In the following, we will discuss only the networks trained with random patterns. We tried many runs from different initial conditions. The number of neurons was varied, but we tried to have the number of hidden neurons as small as possible. The algorithm failed to converge when the number of neurons in the output layer was bigger than 2. The increase of the number of neurons in the hidden layer did not help the convergence. In the cases where the algorithm converged the maximum number of iterations was 20000.

4.6 The analysis of the successful configurations

When the output units of the network were Gaussian velocity detectors, the configuration was considered successful if each output unit had the mean-square error less than 0.2. In other cases we considered as a good configuration the networks whose performance was more than 90% of correct identifications for each output unit. At the appropriate velocity, a response of the output unit was considered correct if it was larger than 0.7. Otherwise a correct response had to be smaller than 0.3. In both cases the performance was measured using a set of patterns different from the training set.

The number of possible network configurations is very large. Different solutions all look different. Two typical examples of networks with one output neuron are shown in
Fig. 4.5, with the corresponding velocity discrimination curves shown in Fig. 4.7. In these examples, the network comprised 5 input components and 5 hidden neurons. The training set consisted of 2000 patterns moving with a velocity between -2 and 2. The target velocity is 1. It is interesting that several independent configurations can be used together to improve the performance. For example, the two networks of Fig. 4.5 will both give an incorrect answer in only 1% of all presentations. This result is typical and indicates a large degree of independence between the solutions. In Fig. 4.6 is represented a successful configuration of the network with one output neuron, Gaussian velocity detector. Its output is 1.0, 0.6, 0.14, and 0.012 when the pattern velocities are 1, 0, -1, and -2 respectively.

Although apparently different, the solutions share some common features. In all successful configurations, with one or two output neurons (velocity detectors) there are two or three hidden units where the connection strength for direct and delayed inputs are inverted in sign and shifted in space by an amount corresponding to the target pattern velocity of the detector. The combined effect of those units is silence if the pattern moves across the field with the appropriate velocity, and inhibition of the corresponding output unit otherwise. In Fig. 4.5 such hidden units are (1a), (5a), (2b), and (5b). When the pattern moves with the appropriate velocity 1, their response is 0. In this case at least one of other hidden units will have the response 1 that is enough to cause the output neuron to have response 1. When the pattern moves with a velocity different from the appropriate one, some hidden inhibitory units will respond and inhibit the corresponding output unit. While inhibitory links are a common ingredient of neural network models, here we observe specific inhibitory units which are constructed in accordance with a well-defined set of rules. In Fig. 4.6 such units are 1 and 4.
Figure 4.5  Two typical examples of constructed networks are shown in part (a) and (b) of the Figure. In each part, the connection strengths of hidden units for the direct inputs are shown in the first row. The connection strengths for the delayed inputs are shown in the second row. The connection strength between the hidden and output units are shown at the right side of the figure. The bottom row shows thresholds levels for all units. The thresholds are shown on a scale between 3 and -3 while the absolute value of the largest strengths is equal to 16. The response of the output unit is 1 when the heterogeneous pattern moves with velocity 1 and 0 otherwise.
The mechanism is highly reminiscent of inhibition found in motion-sensitive simple cells in the area V1 of the visual cortex of cats [Maffei et al., 1976]. In those cells, the unresponsive inhibitory zone is a separate mechanism extending over a region larger than the excitatory receptive field centre. A stimulus moving in the null-direction anywhere within the unresponsive inhibitory zone will suppress the cell's response.

In some solutions we noticed another rule reminiscent of some biological systems. In these configurations there are two inhibitory hidden units corresponding to the same output unit with almost the same connection strengths but reversed in sign. A perfect example of this symmetry are units (2b) and (5b) in Fig. 4.5b. Neighbouring simple cells with dark- and light-sensitive regions (indicated by a 180-degree shift in the phase of the response to a drifting grating) have been observed in the cat visual cortex [Foster et al., 1983]. What is the function of this rule? The inhibitory hidden units have the response large than 0.5 when the linear combination of its inputs is larger than threshold.

Figure 4.6 A typical example. The response of the output unit is an approximate Gaussian function of image velocity with the peak at the velocity 1.
level. For same patterns moving with the velocity other than the appropriate one the linear combinations of inputs may be smaller than the threshold level. At that velocity, at least one of the inhibitory hidden units should have high response in order to inhibit the corresponding appropriate velocity detector. With only two units having the connection strengths inverted in sign the whole range is covered.

![Graph](image)

**Figure 4.7** Discrimination of different pattern velocities by several specific configurations of described neural networks. The average response of the output unit averaged over 300 pattern presentations is plotted as a function of stimulus velocity. ——— and ———, networks shown in Figs. 2(a) and 2(b) respectively. ———— network shown in Fig. 4.8 constructed using the acquired heuristics.

3The values around zero correspond to homogeneous patterns.
Extracting velocity of images is important not only for biological systems but for artificial systems as well [Keirstead and Huberman, 1986]. Having several detectors to cover a scope of measured variable may seem as inefficient use of resources. It is interesting that recent approach to implement the computation of image velocity in real-time was based on similar principles [Bulthoff et al., 1989]. The algorithm is computing the autocorrelation of the image with its displacement corresponding to different images.

4.7 Constructing a large network using heuristic rules

In the previous section we described the common features of the set of successful configurations. We wanted to use this knowledge to construct large networks that could not be constructed by back-propagation. The knowledge we obtained can be described in a more formal way as:

a) per each output neuron (velocity detector) at least two hidden neurons functioning as inhibitory units are needed;

b) if only two hidden neurons, \( j \) and \( k \), are used per velocity detector, the connection strength of this two units has to be inverted in sign

\[ w_{ji} = -w_{ki} \]

c) connection strengths of each hidden unit inhibitory unit \( j \) are such that time-delayed inputs are inverted and shifted in space by the amount...
corresponding to the velocity selected as the output

\[ w_j(i + v + I) = -w_{ji}, \quad 0 \leq i < I - v \quad \text{when} \quad v \geq 0, \]

\[ v \leq i < I \quad \text{when} \quad v < 0, \]

\[ w_{ji} = 0 \quad \text{otherwise}; \]

where \( I \) is the number of input components.

d) the connection strengths between the hidden and output units are negative for corresponding inhibitory units, positive otherwise

\[ w_{ji} < 0 \quad \text{when} \quad j\text{-th hidden unit is inhibitory unit for} \ i\text{-th output unit}, \]

\[ w_{ji} > 0 \quad \text{otherwise}; \]

e) the thresholds levels of inhibitory and output units are negative

\[ \theta_j < 0. \]

The rule (a) gives the estimate of the lowest number of hidden units. The number of output units is known by definition. The rules (d-e) give the estimate of the scope of parameters. This rules and simple functional relationships given by rules (b) and (c) makes the search as optimisation with constraints.

We searched for the configuration of a large network with two hidden inhibitory units per each output unit. We used the following procedure:

step 1: the connection strengths of hidden units are determined randomly, according to rules (b) and (c);

step 2: the remaining parameters were determined by the back-propagation,
where the initial conditions were selected according to the rules (d) and (e).

The procedure would be repeated until satisfactorily configuration was found. To achieve convergence it was only necessary to start the search from the initial conditions selected according to the rules (d) and (e). We did not need to have constrained optimisation. That means that the initial conditions were in the basin of the attraction of the successful configuration.

Figure 4.8 The network constructed using acquired heuristics. Each neuron in the output layer is a velocity detector. The appropriate velocities of output neurons are -2, -1, 0, 1, and 2 starting from the left in figure.

With this approach the number of parameters was drastically reduced. Also, the initial condition is closer to the solution state. There was no problem in constructing a network with tens of output units. In Fig.4.8 is shown one configuration determined in this way. The velocity discrimination curve of the output neuron with the appropriate velocity 1 is represented in Fig. 4.7.
4.8 Discussion

While we tested the approach on a particular example, we believe that it is rather general, as indicated by the analogy with the artificial intelligence. Even in the framework of the back-propagation algorithm the approach is not restricted to the composition of sigmoid basis functions. One can take as basis functions radial basis functions or splines that are more useful for constructing nonlinear models [Moody, 1988; Casdagli, 1989].

The necessary requirements for the application of the approach to approximating nonlinear mappings are that the similar simple mapping can be found and that the approximation of simple mapping can be analysed. While the second requirement is usually satisfied, the first requirement may be a problem. In our case, it was obvious what is the simpler mapping. When one is constructing a nonlinear model from data, usually there is no a priori knowledge about the mapping. How to find a similar simple mapping in such a case?

We will only suggest a possible solution to this problem. Instead of approximating the whole function at once we can first approximate the function in a part of its domain. Natural candidates for the parts would be linear subspaces of the domain. For example, when the domain is a two-dimensional space one can try approximations along the lines in the domain. Successful approximations would be analysed. After sampling the function in a sufficient number of different regions we may acquire some heuristic knowledge about what are suitable basis functions or how noisy the data are. We believe that in this way much of computational time may be saved.
Some aspects of processing temporal inputs with cellular automata

A common feature of natural processing systems is that their functioning is a result of the functioning of many simpler interconnected dynamical components. For example, as a simpler unit of the neural system one can consider a part of a dendritic branch, a whole neuron or a neural network. The activity of one unit is dependent upon the activity of neighbouring units. The functional connections between units can be changed during the system’s activity. The dynamical behaviour of a neural system is so important that Maturana [1987] even argues that the system should be studied primarily as a nonlinear dynamical system continuously interacting with the environment.

Cellular automata are a class of mathematical models of a system composed of many simple identical components acting together. They were introduced by von Neumann and Ulam [Codd, 1968] as abstract models of biological self-reproduction. Since then, cellular automata were used in a wide variety of contexts\(^1\) [See Demongeot \textit{et al.}, 1985].

\(^1\) "They (cellular automata) have been the topic or the excuse for countless doctoral theses." [Toffoli and Margolus, 1987].
We investigated how to process temporal inputs with one-dimensional cellular automata. The usual approach to processing with cellular automata is to take an input as the initial state of automaton [Wolfram, 1986], that is, considering automata as a closed dynamical system. In order to increase the similarity between automata and natural processing system, we investigated a class of cellular automata that continuously interact with the temporal input. Their state is not only dependent on a previous state but on temporal input as well. Our goal was to investigate a potential relationship between the pattern generated by the evolution of such automata (a trajectory) and the temporal input pattern.

The input to the system consisted of one-dimensional binary patterns. Mainly, the patterns were objects (strings of 1’s and 0’s) moving with uniform velocities in front of homogeneous background (strings of 0’s). We found that some automata evolve to a heterogeneous pattern only if the ‘object’ moves with a particular velocity or a range of velocities. This ‘filtering of velocities’ effect can be useful for detection of changes in an input. The filtering effect is robust with respect to the noise in input. The effect can be destroyed by having unreliable components of automata or having more complicated input.

In the first sections of this chapter we will briefly describe one-dimensional cellular automata. The automaton constructed by Jen [1986a] for filtering binary strings will be mentioned in the section 5.2. In the third section we will describe the system which Keirstead and Huberman [1986] constructed for detecting the motion of one-dimensional patterns and explain our system. The phenomenon of filtering velocities will be shown in the following section. We will explain the mechanism of the filtering in the section 5.5 and in the last section we will show how the effect breaks down.
5.1 Cellular automata

Cellular automata are spatially extended discrete dynamical systems. They can be thought of as a regular array of simple identical processing units. Each unit takes on \( k \) possible values. The global evolution of cellular automata is determined by local interaction between neighbouring units. We were investigating one-dimensional cellular automata. The value \( a_{t,i} \) of the unit at each position \( i \) depends on the last values in a neighbourhood of at most \( 1 + 2r \) units, \( r \geq 0 \). It is updated in discrete time steps according to a rule \( f \)

\[
a_{t+1,i} = f \left( a_{t,i-r}, a_{t,i-r+1}, \ldots, a_{t,i}, \ldots, a_{t,i+r} \right).
\]  (5.1)

Even the simple cases of one-dimensional cellular automata exhibit a diversity of behaviour, some of which can be very complex. The units of the 'elementary' cellular automaton [Wolfram, 1984] can take only two values, \( k = 2 \). The interaction between units is restricted to the nearest-neighbours, \( r = 1 \). There are 256 different rules \( f \) of the elementary cellular automaton that can be labelled in the following way

\[
f = \sum_{i=0}^{7} \beta_i 2^i,
\]  (5.2)

where \( \beta_i \in \{0, 1\} \) is the value assigned to each of the 8 possible configurations of neighbouring units

\[
a_{t,i-1}, a_{t,i}, a_{t,i+1}
\]

\[
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow
\]

\[
a_{t+1,i}
\]

\[
\beta_7 \quad \beta_6 \quad \beta_5 \quad \beta_4 \quad \beta_3 \quad \beta_2 \quad \beta_1 \quad \beta_0
\]
The evolution of another simple case of one-dimensional cellular automata is governed by the ‘totalistic’ rule [Wolfram, 1984]

\[ a_{t+1,i} = f(a_{t,i-2} + a_{t,i-1} + a_{t,i} + a_{t,i+1} + a_{t,i+2}) \]  

(5.3)

This rule can be also labelled according to the code (5.2) where the index \(i\) of the \(\beta_i\) is the sum of values of neighbouring sites, \(i \in \{0..5\}\).

We will consider the automata whose units are connected in a ring (Fig. 5.1a). The evolution of one-dimensional cellular automata can be easily visualised. Fig. 5.1b shows the spatio-temporal pattern that is the trajectory of an elementary automaton, rule \(f = 54\), starting from the ‘disordered’ initial state. In the initial state each unit value is independently chosen as 0 or 1 with a probability of 0.5.

**Figure 5.1** a) The schematic representation of a one-dimensional cellular automaton whose \(N\) units are connected in a ring. b) The spatio-temporal pattern representing the evolution of the automaton. Black squares represent units with value 1; white squares units with value 0. Each row is the state of the automaton at time \(t\). The initial state is the first row.
The spatial and temporal sequence generated by the evolution of one-dimensional cellular automata can vary in their form from regular to seemingly random patterns. Based on systematic computer simulation Wolfram [1984] classified automata in four main classes according to the corresponding spatio-temporal patterns. When the initial state is disordered each class can be described by:

**class 1**: evolves to the homogeneous pattern;

**class 2**: evolves to a set of separated simple stable or periodic structures;

**class 3**: evolves chaotically;

**class 4**: evolves in a complicated and unpredictable way.

![Figure 5.2 Examples of the four classes of automata behaviour. The first three patterns are generated by elementary automata: a) class 1, \( f = 168 \); b) class 2, \( f = 29 \); c) class 3, \( f = 22 \); d) class 4; the pattern is generated by the totalistic rule \( f = 52 \).](image-url)
Patterns corresponding to these classes are illustrated in Fig. 5.2. The evolution of two-dimensional cellular automata can be classified in the same way [Packard and Wolfram, 1985]. In addition to qualitative classifications, each behaviour can be characterised by measuring entropies, dimensions, information flow, or inferring the grammar of the system.

Detailed mathematical understanding of cellular automata is restricted to the most simple cases [Jen, 1986b]. For example, even the evolution illustrated at Fig. 5.2d is undecidable [Wolfram, 1985]. There is no way to predict what the state of the system will be after several time steps, except to do the exact simulations. The investigation of cellular automata relay heavily on extensive computer simulations. The rules and initial states are heuristically chosen and the evolution simulated. Once, the interesting phenomena are identified, the system may be investigated in detail using some analytical approximation.

5.2 One-dimensional cellular automata as filters of binary strings

From the beginning of their introduction, cellular automata were viewed as processing systems. Von Neumann has shown that some two-dimensional cellular automata are equivalent to a universal Turing machine [Codd, 1969]. Even some classes of one-dimensional cellular automata are computationally universal [Smith, 1971]. The initial state of a cellular automaton represents a ‘program’ and an ‘input’. The ‘input’ is processed by the cellular automaton evolution until a state representing the ‘output’ is reached. It is interesting that some commercial highly parallel computers are based on cellular automata [Hillis, 1984].
As a particular example of processing with cellular automata we will mention Jen’s approach [Jen, 1986a] to filtering binary strings. During the evolution of some elementary cellular automata, part of the input state remains invariant. Jen calls rules of such automata ‘filtering’ rules. In Fig. 5.3a is shown the evolution governed by such a rule. The invariant string (sequence of 0’s and 1’s) is independent of string spatial positions or the values of its neighbouring units.

![Figure 5.3](image)

**Figure 5.3** a) During the evolution of elementary cellular automaton \( f = 12 \) the marked parts of the initial state remain invariant. b) The evolution of an elementary automaton whose units take on three values. Gray squares represent units with third value \( _ \), ‘blank’. The values of units are updated according to rule: \( \{001, 100\} \rightarrow 0; \{010, 011, 110, 111\} \rightarrow 1; \{ _10, 11_{_}\} \rightarrow 1; \) for all other tuples the assigned value is \( _ \). After Jen [1986a].

Jen gave the procedure for finding the rule under which a given arbitrary string is invariant. She showed how to extend an elementary cellular automaton by allowing units to take three values. The initial state of the automaton is a binary string. After several iterating steps only the invariant parts of the initial state remain. The other parts of the initial state are filtered, with the corresponding units taking the third value ‘blank’ (Fig. 5.3b).
5.3 Processing temporal inputs with the cellular automata

The straightforward approach to processing temporal inputs would be similar to our usage of a neural network to extract the velocity of moving images. We can take several delayed states of the input as the initial state of one- or two-dimensional cellular automata and the processing is the same as the processing of static data. The rule of the automaton should be selected so that attracting states correspond somehow to the desired output. Another approach would be to use a cellular automaton whose state is dependent on both previous state and time dependent input. Let us first, describe a similar approach to processing temporal inputs proposed by Keirstead and Huberman [1986], because it performs the same task as our system.

Their goal was to construct a processing system which can detect spots moving in one dimension, one site per time step in the presence of noise. Like one-dimensional cellular automata the system they constructed consists of a one-dimensional array of processing units. Only nearest units are interconnected. Each unit receives time-dependant binary input and is characterised by two memory values $R_{t,i}$ and $L_{t,i}$ which are updated every time step according to

\begin{align*}
\text{if } x_{t,i} = 1 \text{ then } & R_{t,i+1} := R_{t-1,i} + (1 - x_{t-1,i}), \\
& L_{t,i+1} := L_{t-1,i} + (1 - x_{t-1,i}), \\
\text{if } x_{t,i} = 0 \text{ then } & R_{t,i+1} := R_{t-1,i} - 1, \\
& L_{t,i+1} := L_{t-1,i} - 1, \\
\text{if } R_{t,i} > S \text{ then } & R_{t,i} := S, \\
\text{if } L_{t,i} > S \text{ then } & L_{t,i} := S,
\end{align*}

(5.4)
where \( x_{t,i} \) are the values of input at position \( i \) and at time \( t \) and \( x_{t,i} \in \{ 0, 1 \} \), \( R_{t,i}, L_{t,i} \in \{ 0, \ldots, S \} \), \( S \) is some integer. To check whether the spots are moving left or right is to check if the memory values \( R_{t,i} \) and \( L_{t,i} \) are bigger than some threshold. The main property of the system (5.4) is its ability to accurately detect motion in the presence of noise\(^2\).

The system we investigated is simple. We took the elementary cellular automaton and added time-dependant input to each unit. The values of units are updated according to the following rules

\[
\begin{align*}
\text{if } x_{t,i} = 1 \text{ then } & a_{t+1,i} = f_1 (a_{t,i-1}, a_{t,i}, a_{t,i+1}), \\
\text{if } x_{t,i} = 0 \text{ then } & a_{t+1,i} = f_0 (a_{t,i-1}, a_{t,i}, a_{t,i+1}),
\end{align*}
\]

(5.5)

where the time-dependent input \( x_{t,i} \) is defined as in (5.4), and \( f_1 \) and \( f_0 \) are labeled using the code (5.2). When the input is homogeneous, the evolution of the system (5.5) is the same as the evolution of the elementary automata. When the input is static, the system (5.5) is equivalent to multiple-scale cellular automaton introduced by Wolfram [1986] to increase pattern recognition capabilities of cellular automata and model hierarchical systems\(^3\).

We mainly investigated the behaviour of the system (5.5) when the input consisted of objects moving with uniform velocities. In case of one dimension and a binary input we consider as an object a string of 1's and 0's, where the number of 1's is bigger than the number of 0's, and the length of the object is smaller than the length of the automaton.

---

\(^2\) With 15% noise in the input, the probability of accurate detection is 90%. Noise in the input means that there is some probability that the value of the elements of the input can be flipped into opposite state.

\(^3\) This multiple-scale cellular automaton consists of two automata evolving on different time scales. The local transition rule for a unit of the faster automaton is a function of the value of the corresponding unit of slower automaton.
Figure 5.4 Some examples of the behaviour of the systems (5.5). The systems evolve from the disordered initial state. a) The input pattern that is a string of ones of length 21 shifted to the right one site per time step, that is with velocity 1. b) The evolution of the system with rules $f_0 = 40$ and $f_1 = 22$; the input as represented in a). c) The evolution with rules $f_0 = 18$ and $f_1 = 105$; the input as in a). d) The same as b) except the initial state. e) The input pattern; velocity -1. f) The evolution under the same rules as in c) but with input as in e).
This class of inputs can be visualised as black objects or black objects with white stripes moving in front of the white background (Fig. 5.4 a and e). Some examples of the behaviour of the system (5.5) are shown in Fig. 5.4.

The information processing with the system (5.5) can be thought of as mapping of spatio-temporal input pattern to spatio-temporal output pattern that is the trajectory of the cellular automaton. Like the approach of mathematical morphology to image analysis [Haralick et al., 1987], we can use the output pattern as an input for the next stage of processing. The initial pattern can be transformed with the cellular automaton so that only relevant information is preserved, which makes next stages of processing easier.

The output pattern is dependent on the initial state of the automaton (Fig. 5.4 b and c). While sometimes it is an undesirable feature, it can be used for context dependent processing. The mapping between input and output patterns can be many-to-one. For example the output patterns for the cellular automaton illustrated at Fig. 5.4 c are statistically similar for different initial states and different objects moving with the same velocity. Using this behaviour for processing would be equivalent to the ‘computing with attractors’. Another useful property of the system (5.5) is that global properties of the output pattern can be dependent on local properties of input pattern.

5.4 Filtering of the velocities

As can be seen from examples in Fig. 5.4 the output pattern is preserving some features of the input pattern. An interesting phenomenon can be observed when the rule $f_0$ of the system (5.5) belongs to a class of rules for which the elementary cellular automaton evolves to the zero state (all units with value zero) after several time steps (Fig. 5.2a). An example of the evolution of the system with such a rule is shown in Fig. 5.4 b and
d. After several time steps only the units where the object is, or the units close to the object, may have the value 1. The input pattern is mapped into a similar output pattern. The output pattern is sharply dependent on the velocity of the object. For some rules $f_i$ the system will evolve to zero state after several time steps for all object velocities, except one velocity or a narrow range of velocities. We can consider such a behaviour as the 'filtering' of velocities. The phenomenon is reminiscent of Jen's filtering of binary strings. It can be used for achieving the same functionality as that of the Keirstead and Huberman system.

**Figure 5.5** The example of ‘filtering’ velocities. When the rule $f_0$ is 128 and $f_1$ is 108 the system (5.5) evolves to the zero state for all velocities different from zero. The initial state is disordered. The shape of the moving object is the same as shown in Fig. 5.4a a) velocity = 0. b) velocity = 2. c) velocity = -0.5. d) velocity = 1.
For example, for some rules $f_1$ the output pattern will be created only when the object is static (Fig. 5.5). We say that there is no output pattern if the system evolves to zero states. For some rules $f_1$ pattern appears only if the object moves with the velocity 1 (Fig. 5.6a), and for some rules $f_1$ the only velocity that is not filtered is -1 (Fig. 5.6b). Using a cellular automaton with the range $r$ larger than 1 velocities higher than 2 or -2 can be filtered. For some rules the pattern is independent of the velocity of inputs. The system whose behaviour is shown in Fig. 5.4 b and d responds in the range -1 to 1.

a) ![Image](image1.png)

b) ![Image](image2.png)

**Figure 5.6** An example of ‘filtering of the velocities’. The initial state is disordered. The input is the same as shown in Fig. 5.4a a) When the rule $f_0$ is 128 and $f_1$ is 152 the system (5.5) evolves to the zero state for all velocities different from 1. b) The velocity -1 is not filtered out when $f_0 = 128$ and $f_1 = 74$.

In general, the filtering effect is not affected by a reasonable level of the noise in the input pattern. The effect of the high level of noise is that the system may evolve to the zero state even when the objects move with the velocity that is in the unfiltered range. The level of noise which can cause the failure of the system depends on particular combination of $f_0$ and $f_1$, but it is always possible to find the combination of rules such that 10% of noise is not destroying the filtering effect. Fig. 5.7 shows the probability that the systems illustrated in Fig. 5.4b and Fig. 5.5 will not evolve into the zero state when the noise is present.
Figure 5.7 The probability that the system (5.5) do not evolve into zero state when the velocity is correct and the noise is present. The curves are obtained for the velocity \( f_0 = 128 \); \( f_t = 108 \); \( f_t = 22 \).

5.5 The mechanism of filtering

To explain the mechanism of filtering velocities with a cellular automaton we will think of an automaton as an excitable medium. The units with the value 1 will be considered as ‘excited’. The initial state that has all units with the value 0, except for a few nearby units, will be considered as the initial excitation, and the evolution of an automaton can be seen as the propagation of the excitation. Numerical simulations of one-dimensional cellular automata \( (k = 2, r = 1, 2) \) suggest that they can be classified in five classes according to the way the excitation propagates. In the course of the evolution, initially localised excitation:

- **class A**: disappear with time;
- **class B**: are sustained but do not propagate;
**class C:** propagate in a particular direction with a fixed speed;  

**class D:** propagate to all directions with a fixed speed;  

**class E:** propagate and contract irregularly.

These classes are illustrated in Fig. 5.8. The classes are congruent with previous classification of cellular automata according to the evolution, except that the class 2 is refined with classes B and C. The maximum possible speed of propagation of an excitation of a given class of automata is equal to $r$.

**Figure 5.8** Examples of the propagation of the excitation. The first four patterns are generated by elementary automata: a) class A, $f = 168$; b) class B, $f = 29$; c) class C, $f = 38$; d) class D, $f = 22$; e) class E, the pattern is generated by the totalistic rule $f = 20$.

The class of rules $f_0$ that we selected does not allow the initial excitation to propagate. When the system (5.5) exhibits the filtering behaviour the only areas where the
excitation can propagate are the units where the object is located. The object has the role of the guide for the propagation. If the direction of the object and the direction of the propagation of the excitation do not coincide, the output pattern is not created.

Knowing the mechanism of this phenomena it is easy to find a rule to achieve a desired behaviour by simulation. One has to search for the rules for which the excitation propagates in the desired way. For the elementary cellular automaton the desired rules can be found analytically, but when one considers noise in the input the analysis becomes very hard.

5.6 Breakdown of filtering

The mechanism of the filtering of velocities is the inability of the initial excitation to propagate. The interesting behaviour can be observed when the units of the automaton are unreliable. Occasionally, a unit can flip into the opposite, erroneous value. The unreliable components produces a ‘seed’ excitation which can cause the appearance of the output pattern. A typical result of a simulation is presented in Fig. 5.9. A small probability \( p \) that the unit flips into an erroneous value causes partial generation of the output pattern that, however, is not sufficiently stable for its precise characterization. For the probability of about \( p = 0.01 \) the generated pattern is easily distinguishable. High probability destroys the structure of the output pattern. There is a qualitative similarity between this phenomenon and the phenomenon of self-organization in synergetic systems through transients, where some fluctuations must be present [Haken, 1983].
Figure 5.9 The effect of unreliable components on filtering. The input pattern is shown in Fig. 5.4 a, except that here the velocity is 2. $f_0 = 40$ and $f_1 = 22$, the initial state is disordered. a) Probability $p$ that the unit flip into erroneous value is $p = 0$; b) $p = 0.005$; c) $p = 0.01$; d) $p = 0.05$.

Another source of the excitation can be the background or an object moving with a correct velocity. The output pattern when an object passes across stationary stripes, i.e. the 'background' is shown in Fig. 5.10. If one is interested in detecting the relative velocity of objects, this phenomena can be useful.
Figure 5.10 The effect of static background to filtering. $f_0 = 40$ and $f_1 = 22$, the initial state is disordered. 

a) The output pattern when the input is like the one shown in b) except for the vertical line. 

b) An input pattern. 

c) The output pattern when the input is like the one shown in b). 

d) The output pattern when there are more lines in the input.
5.7 Conclusion

We do not see the study of cellular automata as important for the direct practical applications in information processing. Rather, it is a simple way for obtaining experience in processing with highly parallel systems. After our experience with automata we feel uncomfortable about a methodological problem. What is a good way other than trial and error, to understand systems like cellular automata? There was no systematic way to search for the interesting behaviour. Because of the system size, in our case the undecidability was not a problem. The exact simulation was easy so we could freely search for the desired behaviour. Because of the simplicity of the system, it was easy to explain the discovered phenomena. But in the case of real distributed systems this approach is not practical. An obvious approach would be to derive a simpler model that could be directly simulated. As another approach, we expect that the problems could possibly be treated by the development of computer tools for monitoring such systems.
REFERENCES

Aizawa K., Harashima H, and Saito T. [1989], Model-based analysis synthesis image coding (MBASIC) system for a person’s face, Signal processing: Image communication 1, 139.


Berge P., Pomeau Y., and Vidal C. [1986], Order within Chaos, Herman.

Brent R. P. [1990], Fast training algorithms for multi-layer neural nets, Numerical Analysis Project, NA-90-03, Computer Science Department, Stanford University


Casdagli M. [1989], Nonlinear prediction of chaotic time series, Physica D 35, 335.

Chillingworth D.R.J. [1976], Differential topology with a view to applications, Research Notes in Mathematics 9, Pitman.


Collet P. and Eckmann J.-P. [1980], Iterated maps on the interval as dynamical systems, Birkhauser.


Crutchfield J.P. and McNamara B.S. [1987], Equations of motion from a data series, Complex Systems 1, 417.


Farmer J.D. [1982a], *Information dimension and the probabilistic structure of chaos*, Z. Naturforsch 37a, 1304.

Farmer J.D. [1982b], *Chaotic attractors of an infinite-dimensional dynamical system*, Physica D 4, 366.


Fraser A.M. [1989b], *Reconstructing attractors from scalar time series: a comparison of singular system and redundancy criteria*, Physica D 34, 391.


Gibson J.J [1979], *The ecological approach to visual perception*, Houghton Mifflin.


Haken H. [1983], *Advanced synergetics*, Springer-Verlag.


Hewitt C. and de Jong P. [1982], *Open systems*, AIM 691, MIT


Holland J.H. [1975], Adaptation in natural and artificial systems, University of Michigan Press.


Huberman B.A. [1985], Dynamics of computing structures, Physica Scripta T9, 165.


Kostelich E.J. and Yorke J.A. [1990], Noise reduction: finding the simplest dynamical system consistent with the data, Physica D 42, 183.


Mackey M.C. and Glass L. [1977], Oscillation and chaos in physiological control systems, Science 197, 287.

Maffei L. and Fiorentini A. [1976], The unresponsive regions of visual cortical receptive fields, Vision Res. 16, 1131.

May R.M [1976], Simple mathematical models with very complicated dynamics, Nature 261, 459.


Mees A.I. [1989], Modeling complex systems, Research Report, Dept. of Mathematics, Univ. of Western Australia.


Pearl J. [1985], Heuristics: intelligent search strategies for computer problem solving, Adison-Wesley.


Rosen R. [1985], Anticipatory systems, Pergamon Press.


Ruelle D. [1989], Chaotic evolution and strange attractors, Cambridge University Press.


Tesauro G. and Sejnowski T.J. [1989], *A parallel network that learns to play backgammon*, Artificial Intelligence 39, 357.


Werbos P.J. [1988], *Generalisation of backpropagation with application to a recurrent gas market model*, Neural Networks 1, 339.

